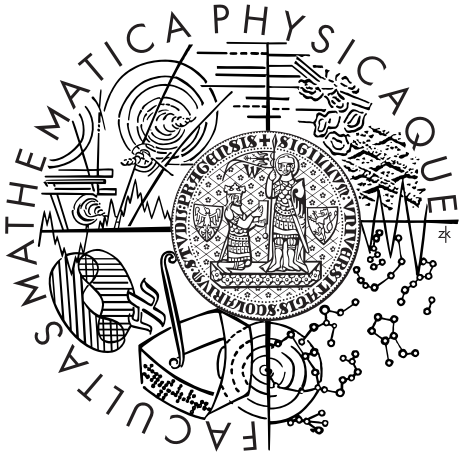
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Šebesta

Wave editor

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D.
Studijní program: Informatika
Studijní obor: Programování

Praha 2011

Děkuji svému vedoucímu RNDr. Janu Kofroňovi, Ph.D, za pomoc při tvorbě této bakalářské práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....

Název práce: Wave editor

Autor: Michal Šebesta

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D, Katedra distribuovaných a spolehlivých systémů

Abstrakt: Cílem bakalářské práce je vyvinout aplikaci pro práci se zvukem. Aplikace se nesnaží konkurovat dlouho vyvíjeným komerčním aplikacím, snaží se pozdvihnout to nejdůležitější pro editaci zvuku, proto obsahuje hlavně ty nejdůležitější a pro práci nejpohodlnější prvky. Těmito prvky rozumíme např. načítání zvukových souborů, nahrávání pomocí mikrofonu, funkce pro stříhání, multitracking a nejrůznější efekty. Pro uživatele je pohodlné i vlastní nastavení, jako jsou klávesové zkratky a design.

Klíčová slova: Editor, audio, zvuk, wave, přehrávač, nahrávač.

Title: Wave editor

Author: Michal Šebesta

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D, Department of Distributed and Dependable Systems

Abstract: The goal of the work is to develop an application for working with a sound. The aplication doesn't try to compete with a long time developed commercial application. It tries to point out what is the most important for sound editing. Therefore the application mainly contains the most important and the most comfortable elements. These elements are e.g. loading sound files, recording with microphone, editing functions, multitracking and various effects. Setting of shortcuts and design is comfortable for users, too.

Keywords: Editor, audio, sound, wave, player, recorder.

Obsah

1. Úvod

Téma této bakalářské práce jsem si vybral proto, protože se rád pohybuji v prostředí hudby. Ať už jako posluchač či muzikant. Jako muzikant jsem pracoval se spousty zvukových editorů, ale žádný se mi nelíbil natolik, že bych v něm dokázal bez výhrad pracovat dlouhodobě. Většinou jsou programy velmi nepřehledné nebo složité na ovládání.

Cílem této práce je tedy vytvořit takový editor, aby bylo co nejpohodlnější v něm pracovat a současně aby poskytoval vše potřebné k editování.

Způsob, jakým jsem řešil, co má v programu být a co má program umět, byl následující. Z vlastních zkušeností vím, co potřebuji, co mi vyhovuje a naopak co mi chybí, když nahrávám a edituji. To mi sloužilo jako základ pro práci. Dále jsem se ptal různých lidí z tohoto oboru, co je nejdůležitější nebo nejpohodlnější pro ně.

Můj program nemůže konkurovat programům vyvíjeným po léta profesionálními vývojáři, to ani nebylo cílem. Nesnažím se udělat lepší editor, snažím se udělat pohodlně použitelný editor.

Kapitola 2 se zabývá popisem pozadí práce – vysvětluje hlavně, co je zvuk v digitální podobě a jakým způsobem je reprezentovaný. V kapitole 3 je provedena analýza práce – výběr jazyka a platformy a jejich možnosti, diskuse návrhu aplikace, rozebrání důležitých algoritmů atd. Kapitola 4 blíže popisuje strukturu řešení programu a kapitola 5 obsahuje programátorskou dokumentaci. Šestá kapitola uvádí existující implementace a srovnává s nimi moji aplikaci. V sedmé kapitole je napsáno shrnutí, přehled toho nejzajímavějšího a možnosti rozšíření.

2. Background

V této kapitole se seznámíme s datovou reprezentací zvukové stopy a vysvětlíme si pár souvisejících termínů. Dále si povíme něco o Fourierově transformaci, jelikož se používá ve spektrální analýze a frekvenčních filtrech. Nejdříve si však ukažme, jak vypadá zvuk v digitální podobě.

2.1 Zvuk v digitální podobě

Zvuk, jakožto spojitý signál, nelze jednoduše reprezentovat v digitální podobě, proto se používá převaděč z analogového signálu na digitální – převádí spojitý signál na diskrétní. Bohužel tím vznikají nepřesnosti, které závisí na parametrech datové reprezentace (objemu vzniklých dat). Tyto nepřesnosti ovlivňují kvalitu výsledného diskrétního signálu. Jak to tak bývá, i zde platí, čím větší datový objem, tím větší kvalita.

2.1.1 Rozlišení

V angličtině „resolution“. Udává nám, jaké hodnoty můžeme použít na zaznamenání amplitudy. Jelikož se pohybujeme ve světě počítačů, je logické, že počet těchto hodnot závisí na tom, kolik bitů pro ně vyhradíme – proto se parametr nazývá „bitů na vzorek“ (v angličtině „bits per sample“). Počet hodnot je tedy nějaká mocnina dvou. Například, pokud budeme mít 8-bitové rozlišení, můžeme zachytit 255 hodnot. Hodnoty pak mohou znamenat rozsah od -128 do 127, což udává důležitost (velikost) amplitudy. Nejpoužívanější rozlišení jsou 16-bitové, 24-bitové a 32-bitové. Pro nejméně přesný záznam pak 8-bitové.

2.1.2 Vzorkovací frekvence

V angličtině „sample rate“ či „sampling rate“. Analogový signál, který je spojitý, je potřebné převést na posloupnost hodnot, které spojité už nejsou. Je potřeba definovat, jak často se mají zaznamenávat jednotlivé snímky. Rychlost, jakou se zachycují snímky, se nazývá vzorkovací frekvence. Měřítko může být jakékoliv, ale obecně se používá počet vzorků za sekundu.

2.1.3 Zpětné sestavení signálu

Originální signál může být z diskrétních hodnot znovu sestaven pomocí interpolační formule [5] (ta signál zakulatí). Přesnost signálu však závisí na rozlišení a vzorkovací frekvenci.

Rozlišení může způsobit „kvantovací chybu“, což je rozdíl mezi amplitudou originálního a digitalizovaného signálu. Tato chyba se měří v jednotce „nejméně významného bitu“ (v originále „least significant bit“) [6], která vyjadřuje hodnotu posledního bytu (v příkladu 8-bitového rozlišení by LSB bylo $\frac{1}{256}$). Velikost kvantovací chyby je mezi 0 a polovinou LSB, což je logické, poněvadž to je vzdálenost k nejbližšímu číslu, které je na stupnici rozlišení.

velikost	popis	hodnota
4	ID části	„RIFF“ (0x52494646)
4	velikost	velikost souboru – 8
4	RIFF typ	„WAVE“ (0x57415645)
ostatní části		

Obrázek 2.1: Hlavička souboru

Zapomeňme pro chvíli problém daný rozlišením. Přesné sestavení originálního signálu je možné, pouze pokud je vzorkovací frekvence vyšší, než dvojnásobek nejvyšší frekvence v digitálním signálu. O tomto problému pojednává „Nyquist sampling theorem“ [8].

2.2 Datová reprezentace zvuku

Pracujeme se zvukem ve formátu „Wave“, což je pro OS Windows přirozený formát pro uchování dat digitálního zvuku. Je to jeden z nejpodporovanějších zvukových formátů na PC díky popularitě OS Windows a mnoha aplikacím napsaných právě pro tuto platformu.

2.2.1 Struktura souboru

Soubory typu „Wave“ podléhají standardům „RIFF“, který dělí obsah souboru na jednotlivé části, jež každá obsahuje svoji vlastní hlavu a tělo. RIFF struktura je pohodlná v tom, že dovoluje nepoužívat všechny části. Je vhodné podotknout, že veškerá data jsou ukládána v Little-Endian pořadí.

2.2.2 Hlavička souboru

Prvních 8 bytů souboru je standardní RIFF hlavička, která obsahuje název (ID) a velikost dané části – pro obojí jsou vyhrazeny 4 byty. ID hlavičky souboru je „RIFF“ a velikost je celková velikost souboru bez velikosti této hlavičky (což je 8 bytů). Potom následují 4 byty, ve kterých je typ RIFF souboru, v našem případě to je „WAVE“. Po této sekci se soubor skládá z jednotlivých již zmíněných částí. Viz obrázek 2.1.

2.2.3 Části souboru

Pro „Wave“ soubory existuje mnoho typů jednotlivých částí. Ačkoliv jich je mnoho, opravdu důležité jsou jen dvě, a to formátová část, která popisuje formát vzorků, a datová část. I když to není obsaženo ve specifikaci, je dobré mít umístěnou formátovací část před vlastními daty. Je tomu tak např. kvůli streamování. Každá část má hlavičku v jednotném tvaru. Hlavička je 8-mi bytová – 4 byty název a 4 byty velikost, kde název se skládá ze 4 písmen. Viz obrázek 2.2.

2.2.4 Formátová část

Obsahuje informace o způsobu vzorkování, typu komprese, počtu kanálů atd. Viz obrázek 2.3.

velikost	popis	hodnota
4	ID části	posloupnost 4 znaků
4	velikost	velikost dat
data		

Obrázek 2.2: Hlavička obecně

velikost	popis	hodnota
4	ID části	"fmt " (0x666D7420)
4	velikost	16 +velikost extra informací
2	kód komprese	0 - 65,535
2	počet kanálů	1 - 65,535
4	vzorkovací frekvence	1 - 0xFFFFFFFF
4	průměrný počet bitů za sekundu	1 - 0xFFFFFFFF
2	zarovnání bloku	1 - 65,535
2	rozlišení	2 - 65,535
extra informace		

Obrázek 2.3: Formátová část

`ID části` — vždy "fmt "(0x666D7420)

`velikost dat formátové části` — standardní velikost (16 bytů) + velikost extra informací

`kód komprese` — 0 pro neznámý, 1 pro nekomprimované PCM, ostatní pro komprimované soubory (např. 2 pro Microsoft ADPCM, 80 pro MPEG)

`počet kanálů` — 1 pro mono, 2 pro stereo, atd.

`vzorkovací frekvence` — počet vzorků za sekundu, tato hodnota není ovlivněna počtem kanálů

`průměrný počet bytů za sekundu` — udává počet bytů, které musí nastreamovat převaděč signálu z digitálního na analogový

`zarovnání bloku` — počet bytů na jeden vzorek (ze všech kanálů)

`rozlišení` — počet bitů na vzorek

`extra informace` — extra informace používané pro kódování (pro PCM neexistuje)

2.2.5 Datová část

Obsahuje vzorkovací data, uloženy podle způsobu komprese. Pokud je komprese 1 (= nekomprimované PCM), jsou ukládány čisté hodnoty.

„Wave soubor“ obvykle obsahuje pouze jednu datovou část, avšak může jich obsahovat i více, pokud jsou obsaženy ve „wavl“ části, což je jakýsi seznam zvukových stop. Viz obrázek 2.4.

`ID části` — vždy "data"(0x64617461)

`velikost dat datové části` — velikost dat

velikost	popis	hodnota
4	ID části	"data" (0x64617461)
4	velikost	0 - 0xFFFFFFFF
vzorkovací data		

Obrázek 2.4: Datová část

`vzorkovací data` — pokud soubor obsahuje více kanálů, jsou do sebe vzorky jednotlivých kanálů „zazipovány“ (Např. pokud uvážíme dva kanály, vzorky půjdou za sebou v pořadí – první vzorek prvního kanálu, první vzorek druhého kanálu, druhý vzorek prvního kanálu, druhý vzorek druhého kanálu, třetí vzorek prvního kanálu atd.)

2.3 Diskrétní Fourierova transformace

Definice. *Diskrétní Fourierova transformace (DFT) je zobrazení* $f : \mathbb{C}^n \to \mathbb{C}^n$*, pro které platí*

$$X = f(x) \equiv X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, ..., N-1$$

, kde `i` *je imaginární jednotka a* $e^{-\frac{2\pi i}{N}}$ *je N-tá odmocnina z jedné [3] [13].*

Je zřejmé, že složitost DFT je $\Theta(n^2)$.

Co je výsledkem DFT? Uvažme vstup jako polynom stupně `n` (posloupnost `n`-1 komplexních čísel s nulovou imaginární částí). Víme, že „*Libovolný polynom stupně (nejvýše) k lze reprezentovat jednak jeho koeficienty, tedy čísly* $p_0, p_1, ..., p_k$*, druhak i pomocí hodnot.*“ [3] a že „*Polynom stupně nejvýše d je jednoznačně určen svými hodnotami v* $d+1$ *různých bodech.*“ [3]. DFT převádí polynom reprezentovaný svými koeficienty na polynom reprezentovaný pomocí hodnot.

Obecněji. „DFT převádí jednu funkci do jiné, o které se říká, že je ve „frequency domain representation“ ke své původní funkci (ta je často v "time domain representation").“ [13]

Pro vlastní implementaci DFT používám algoritmus rychlé fourierovy transformace (FFT) [3]. Ve výsledku to znamená, že je algoritmus rychlejší, avšak dokáže přijímat pouze vstupy s $N = 2^n$.

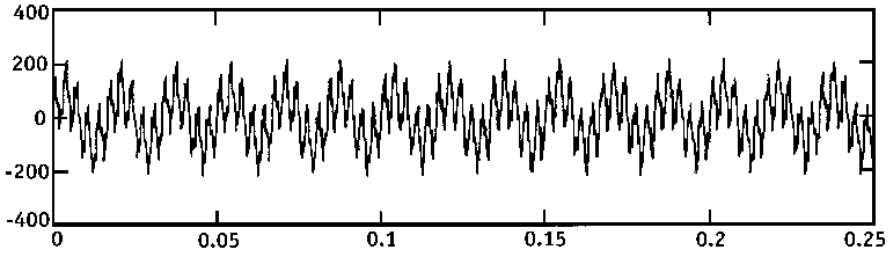
2.3.1 Spektrální analýza

Více o samotném algoritmu pojednává kapitola 3.7.1.

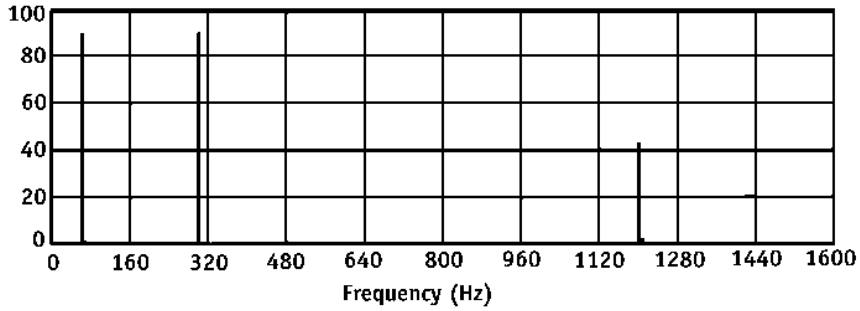
K tomu, abych zpracovávaný signál spektrálně rozložil, používám algoritmus DFT. Zde výsledek DFT interpretujeme jako posloupnost dvojic – amplitudy a fáze k dané harmonické frekvenci. Obráceně, pokud tyto tóny složíme, vznikne nám zpět původní signál. Amplituda značí důležitost dané frekvence a fáze znamená, o kolik je frekvence posunuta před konečným složením.

Z výsledku DFT lze pak snadno sestavit spektrogram, který zobrazuje hodnoty v jednotkách útlumu (decibelech) podle vzorce $L_{dB} = 10 \log_{10} \left(\frac{A_1^2}{A_0^2} \right)$, kde A_0 je referovaná hodnota a A_1 je hodnota měřená [11].

Nejmenší frekvenci, kterou je možné zachytit, je $\frac{vzorkovací\ frekvence}{N}$. Všechny ostatní frekvence jsou pak násobky této „počáteční“. Nejvyšší zachycená frekvence, které se říká „Nyquist frequency“ [9], je polovina vzorkovací frekvence. Polovina je to proto, poněvadž DFT je sudá funkce pro amplitudu a lichá pro fázi. Např. pokud budeme mít $samplerate = 8000Hz$ a $N = 100$, první zachycená frekvence bude $80Hz$, druhá $160Hz$, třetí $240Hz$ atd. Nejvyšší bude $4000Hz$



Obrázek 2.5: Zvukový signál [14]



Obrázek 2.6: Spektrogram [14]

Obrázek 2.5 zachycuje signál složený ze tří tónů, a to o frekvencích $60Hz$, $300Hz$ a $1200Hz$. Jeho snímkovací frekvence je $4096Hz$.

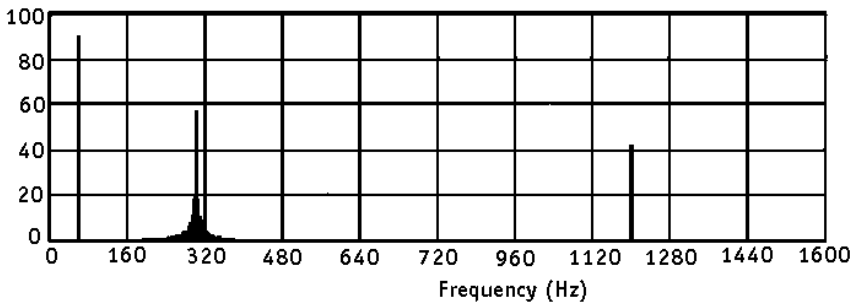
Pokud nasbíráme 1024 vzorků a provedeme FFT, dostaneme odpovídající spektrum, které zachycuje obrázek 2.6.

2.3.2 Leakage

Spektrum předchozího příkladu vypadalo „pěkně“, poněvadž všechny frekvence, které signál obsahoval, byly celočíselným násobkem „počáteční“ frekvence. Jak se ale zobrazí frekvence, které nejsou celočíselným násobkem?

To si názorně ukážeme na následujícím příkladu. Zpracováváme signál stejný jako v předchozím příkladu, avšak tón o $300Hz$ posuneme na $302Hz$. Viz obrázek 2.7.

Můžeme si všimnout, že obrázek kolem $302Hz$ vypadá zcela jinak. Tento úkaz



Obrázek 2.7: Leakage ve spektrogramu [14]

se nazývá „leakage“ [10]. Říká se mu tak, protože to vypadá, jakoby amplituda tónu „unikla“ do okolních tónů. Úkaz nastává tehdy, když signál obsahuje frekvence, které nejsou celočíselnými násobky frekvence „počáteční“.

2.3.3 Window funkce

„Window“ funkce [12] se používají na minimalizaci leakage efektu. Je to posloupnost hodnot od 0 do 1, které nám říkají, jakou váhu má příslušný vzorek. Tato funkce se musí aplikovat na vzorky signálu ještě před tím, než je provedena DFT.

Mezi nejznámější patří obdélníkové, trojúhelníkové, či Hammingovo okno. Obdélníkové okno je posloupnost samých jedniček, což vytvoří identickou posloupnost. Pouze neobdélníková okna mohou omezit „leakage“.

3. Analýza

3.1 Výběr jazyka a platformy

Cílem bakalářské práce je vytvořit pohodlný zvukový editor. To zahrnuje, co nejpohodlnější a nejrychlejší komunikaci mezi uživatelem a programem (GUI). Bylo vhodné volit jazyk z jazyků, které znám a které umožňují požadovanou komunikaci. Ovládám jazyky „C++“ a „C#“. V C++ i C# mohu k co nejjednodušší komunikaci využít rozhraní Windows Forms. C# umožňuje tvořit aplikaci ve WPF. Díky přednostem WPF popsaných v kapitole 3.2 jsem se rozhodl pro variantu WPF v C# a XAML.

Rozeberme si ještě, jestli je to opravdu vhodná volba i z jiného pohledu, než jen GUI.

C# je pro mne přehlednější, nemá dopřednou deklaraci, což je podle mě lepší, a programátor nemusí vlastnoručně spravovat pamět díky „garbage collectoru“. I když jazyk C# by neměl plýtvat s přiděleným procesorovým časem a pamětí, je jasné, že oproti jiným jazykům (např. C a C++) v tomto směru zaostává. Jelikož však netvořím „3D střílečku“, kde bych pracoval s opravdu velkým počtem objektů v co nejmenším čase, a nezáleží mi na každé milisekundě, čas není rozhodující faktor. Tím nechci říci, že je jedno, jak dlouho bude uživatel čekat na zpracování nějaké operace, ale chci tím říci, že jelikož až na výjimky aplikace neprovádí tolik operací najednou, nemůže zapříčinit tak veliké zpoždění, aby to uživatel zřetelně poznal. Co se týče správy paměti, „garbage collector“ odvede všechnu práci. Jistě – může se stát, že tím aplikaci na chvíli pozdrží, avšak četnost je malá, až zanedbatelná.

Snad jediným významným nedostatkem je omezenost na platformy. Jelikož WPF je částí .NET Framework 3.0, aplikaci mohou používat pouze uživatelé Windows Vista a novější, popřípadě Windows XP SP2/SP3 a Windows Server 2003 po instalaci nezbytných knihoven.

Zdálo se, že multiplatformní .NET framework alias „Mono“ by mohl poskytovat vývoj aplikace ve WPF. Ovšem z vyjádření projektu Mono je jasné, že tomu tak není:

„V současné době žádná skupina Mono projektu neplánuje implementovat WPF API jako část svého projektu.“ [19].

Dal jsem přednost WPF před multiplatformním zaměřením.

3.2 Možnosti WPF

Velkou předností WPF je oddělení vlastního kódu od designu, což je důležité pro přizpůsobení vzhledu a jednoduchost ovládání. Další příjemnou vlastností je to, že WPF podporuje starší knihovny, proto je možné spolupracovat s WinAPI a WinForms. WPF renderuje grafiku pomocí Direct3D. To umožňuje hardwarovou akceleraci pomocí grafické karty. WPF umožňuje vytvářet uživatelská rozhraní pro běžná média jako jsou obrázky, audio a video. Podporuje animace. To jsou věci, které mohou zlepšit celkové chování aplikace.

Pozn. I když je WPF vhodné pro renderování grafiky, narazil jsem na problém, když jsem chtěl vykreslovat opravdu veliký počet úseček. Více o tomto problému je

v kapitole 3.7.5.

3.3 Knihovny

K přehrávání a nahrávání zvuků jsem měl na výběr knihovny pod OpenTK a DirectX. Konkrétně, pro OpenTK je to OpenAL a pro DirectX je to DirectSound. Existuje spoustu dalších knihoven, ale tyto jsou nejrozšířenější. WPF poskytuje služby, které umožňují pracovat s běžnými médii, proto za úvahu stálo i nepoužívat žádnou knihovnu a používat zvuky přímo přes WPF. WPF pracuje s DirectX. Jelikož jsem však chtěl tvořit aplikaci, jejíž jádrem je práce se zvukem, bylo vhodné pracovat s něčím na nižší úrovni a tuto variantu jsem zavrhl. Můžeme tedy říci, že jsem se rozhodoval mezi OpenAL a DirectX. Vybral jsem si otevřené OpenAL namísto DirectX od Microsoft proto, že je OpenAL obecně přístupnější a otevřenější.

Pozn. OpenAL je psané v jazyku C++, tudíž je potřeba „wrapper“ na volání funkcí z této knihovny.

3.4 Diskuse návrhu aplikace

Největší dilema bylo, jestli aplikaci separovat na jednotlivé části (část věnující se editaci a část věnující se multitrackingu), jak to je například v aplikaci „Adobe Audition“ [16], nebo jestli mít vše pod jednotnou pokličkou a dělat dohromady. Tak tomu je například v aplikaci „Audacity“ [17]. Obě varianty jsou funkční.

Výhody jednotné verze z hlediska uživatele jsou, že má uživatel vše pohromadě. Vidí jak stopu, se kterou pracuje, tak ostatní, proto může editovat v závislosti na jiných stopách.

Nevýhody jednotné verze jsou naopak ty, že se nemůžeme jednoduše oprostit od ostatních stop. Kdyby se uživatel chtěl soustředit jen na jednu, a to právě editovanou stopu, musel by ostatní nějak schovat, nebo složitěji umísťovat a roztahovat chtěnou stopu přes obrazovku.

V návrhu rozdělených částí stačí překliknou do druhého okna, uživatel se tím oprostí od všeho ostatního a může se věnovat jen dané zvukové stopě. Problém kontextu ostatních stop (editace v závislosti na ostatních stopách) v diskrétním návrhu se dá řešit tak, že si uživatel označí v multitrackingovém okně odkud a kam chce editovat a aplikace přenese označení do editační části.

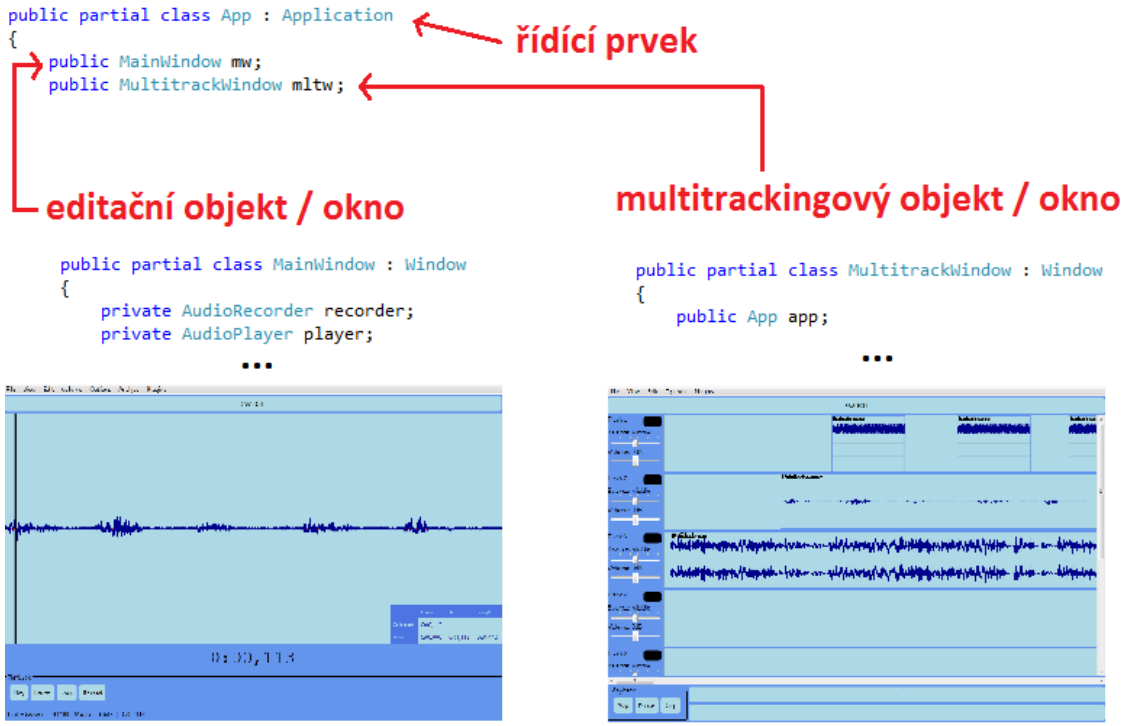
V diskrétním modelu se v každém okně soustředí uživatel jen na to, co chce dělat (editovat, nebo skládat stopy), proto je podle mne tento návrh z hlediska GUI přehlednější.

Z hlediska programátorského je obecně lepší mít vše oddělené.

Po posouzení důvodů z programátorského i uživatelského hlediska jsem zvolil oddělený návrh aplikace.

3.5 Návrh aplikace

Program se skládá ze dvou oddělených velkých částí. To jsou dva objekty, které jsou potomky třídy „Window“ – objekt pro „multitracking“ a objekt pro editaci samotné zvukové stopy (říkejme mu editační objekt). Uvnitř nich probíhají



Obrázek 3.1: Návrh aplikace

všechny nezbytné operace. Z uživatelského hlediska jsou to dvě okna. Nad těmito objekty je postaven řídící prvek, který koordinuje koexistenci a přepínání oken a současně spravuje globální prvky, jako jsou například všechny registrované zvukové stopy nebo kontext knihovny OpenAL či pluginy.

Editační i multitrackingový objekt obsahuje různé funkce na editaci zvukové stopy, vykreslování GUI a update informací, přehrávání, nastavování zkratek a designu atd. Tyto funkce spouští události, které vyvolává především uživatel pomocí tlačítek, klávesových zkratek a podobně. V návrhu GUI bylo tedy zapotřebí zaměřit se na to, co bude uživatel často používat. To navenek zviditelnit více než ostatní – např. tlačítka na přehrávání, menu. Pokusil jsem se uživateli vyjít vstříc ještě víc a implementoval jsem možnost upravit si GUI ke svému pohodlí – toolbary.

Kvůli oddělenému návrhu, každý objekt (okno) odpovídá přizpůsobením GUI a funkcí svému účelu.

Obrázek návrhu aplikace 3.1.

3.5.1 Editační část

V tomto objektu je nejdůležitější komponentou vykreslovací kontrola, do kterého se vykresluje aktuálně zpracovávaná zvuková stopa a která umožňuje editovat svoji stopu.

Editační objekt obsahuje kontroly ovládající zejména editování zvukové stopy, analýzu či přehrávání a nahrávání. Tomu odpovídající jsou i funkce, jež editační okno obsahuje. Dále obsahuje objekt nastavení, který umožňuje nastavit si klávesové zkratky a vzhled. Více v kapitole 4.2.

3.5.2 Multitrackingová část

Tato část slouží pro mixování různých stop dohromady, proto zde objekt neobsahuje editační funkce, ale spíše funkce řídící vykreslování, pohyby zvukových stop, přehrávání, oddělené nastavování stop atd.

Z hlediska GUI k je uprostřed panel určený pro samotný multitracking. Okno se dále skládá z menu a ovládacích prvků, jak to je běžné. Více v kapitole 4.3.

3.6 Pluginy

Měl jsem na výběr, zda-li si vytvořím vlastní interface, nebo použiji nějaký již existující. Jedním z takových je např. VST standard licencovaný jeho tvůrcem „Steinberg Gmbh“. Nemohl jsem použít pluginy z jiných platforem (např. „LADSPA“ pro Unix/Linux). Pluginy psané v jiném jazyku než C# (VST pluginy) je velmi obtížné do aplikace zabudovat. Existuje projekt, který dokáže komunikovat s VST pluginy z jakéhokoliv .NET jazyka, a to „VST.NET“. Jelikož pluginy nejsou hlavním bodem mé práce, rozhodl jsem se nakonec, že si napíši vlastní jednoduché rozhraní i samotné pluginy.

Navrhl jsem aplikaci tak, aby umožňovala hostovat pluginy. Dokáže rozeznat pluginy dvou typů – efektové a designové. Efektové pluginy slouží k úpravě dat zvukové stopy – vytváří efektové objekty. Designové pluginy slouží k definici vzhledu aplikace (barva pozadí, styl tlačítek atd.) – vytváří objekty designu. Oba typy pluginů mají konkrétní interface, přes který komunikují s aplikací. Ve své podstatě mohou pluginy dělat cokoliv, ale aby byly použitelné v aplikaci, musí vytvářet nějaký objekt efektu nebo designu.

3.7 Algoritmy

3.7.1 Spektrální analýza

Problém rozebírá již kapitola 2.3.1.

Cílem tohoto problému je vytvořit spektrogram z posloupnosti vzorků. Základem je udělat DFT. Přesněji, provádím FFT algoritmem „Cooley-Tukey“ [3], který je omezený na vstupy délky 2^n. Analyzuji tedy ne celou stopu, ale pouze její části.

Data určená pro transformaci nejdříve předpřipravíme, a to pomocí „window funkcí“ – tyto funkce ovlivňují spektrální odhad. Pole vzorků vynásobíme příslušnými hodnotami (od 0 do 1).

Provedeme FFT, vezmeme její výstup a uděláme z něj spektrogram. Respektive stačí vzít pouze polovinu výstupu FFT, jelikož to je sudá funkce pro velikosti. Každou hodnotu do spektogramu spočítám podle vzorečku uvedeným v kapitole 2.3.1.

Jediné, co je diskutabilní, je použití FFT namísto DFT. Jelikož složitost FFT je $\Theta(n \log_2 n)$ [3] a složitost DFT je $\Theta(n^2)$ (viz 2.3), vyplatí se využít algoritmus FFT. Omezení na délku vstupu není žádný problém. Za prvé, téměř nikdy není potřeba analyzovat konkrétní velikost, tudíž si přesnou hodnotu mohu zvolit sám. Za druhé, pokud je opravdu třeba použít danou velikost, použiji buffer nejbližší

vyšší přípustné velikosti doplněný nulami, což způsobí jen zanedbatelnou chybu, kterou kompenzuje dosažená rychlost.

3.7.2 Frekvenční filtr

Úkolem algoritmu je odfiltrovat, případně nějakým způsobem hlasitostně upravit, určité frekvence. V programu používám řešení, které využívá FFT pro rozklad na frekvenční spektrum. S tímto rozkladem pak dále mohu pracovat – upravovat hodnoty frekvencí. Dále použiji inverzní FFT, abych získal upravený zvuk.

Problém, který se vyskytuje v důsledku užívání FFT na vstup omezené délky (řekněme např. 1024 snímků), je ten, že na sebe transformované části přesně nenavazují, a proto je ve zvukové stopě slyšet jakési „lupání“. Odstranit toto „lupání“ je složitější. V praxi se tento problém řeší tak, že se využívají „window funkce“. Ty udělají to, že signál na začátku a na konci zpracovávané části utlumí. Funkce by ovšem pouze zapříčinily, že by se zvuk pravidelně a velmi rychle ztišoval a zesiloval. Stopa se proto v praxi nerozkouskovává na pravidelně na sebe navazující úseky, ale i na úseky, které začínají již v nějaké části předchozího úseku a končí v nějaké části následujícího úseku. Tyto jednotlivé úseky se pak mezi sebou průměrují a produkují výslednou stopu. Jednoduše řečeno se vytvoří dvě stopy, které se ztišují a zesilují ve stejném intervalu, avšak tyto výkyvy jsou od sebe posunuty. Stopy se pak složí dohromady, aby hlasitost nekolísala. Tato metoda se nazývá „overlap-save“ metoda [20].

Frekvenční filtr se dá dělat více způsoby – já jsem si vybral způsob přes FFT, protože jsem měl již implementovanou Fourierovu transformaci, kterou jsem mohl využít.

Jiný způsob provedení filtru je například FIR (finite impulse response) filtr, který využívá cyklický buffer na to, aby spočetl výslednou hodnotu pro vstupní vzorek v závislosti na předchozích vzorcích. Výhodou implementace tohoto FIR filtru je ta, že vstup do transformace je vždy jen jeden vzorek a výstup je také jeden vzorek. Filtr se tedy může používat „real-time“ a nemusí bufferovat větší objem dat.

Tento FIR filtr jsem zabudoval do aplikace jako plugin, což je lehce pozměněný kód, jež vytvořil „Neil C, Etanza Systems, 2006“. Kód FIR filtru je volně k distribuci i pro komerční aplikace.

3.7.3 Mixování zvukových stop

V této sekci rozebereme algoritmus pro smíchání více stop do jedné. Nebudu zde řešit, jak mixovat stopy s jinými počty kanálů, ani jak mixovat stopy různých rozlišení či vzorkovací frekvence. Je pouze technická záležitost, jak stopy převést. Berme tedy v úvahu stopy shodující se ve formátu. (Pokud mají více kanálů, mixuji odpovídající kanály zvlášť.)

Když v reálném životě slyšíme více zvuků, slyšíme ve skutečnosti součet všech signálů. Čím více slyšíme zvuků, tím silnější signál přijímáme. Neexistuje žádná horní hranice. Člověk akorát vnímá rozdíl hlasitějších zvuků méně než u slabších. To je také důvod, proč se rozdíl hlasitosti uvádí v logaritmickém měřítku, aneb decibelech.

Ve zpracování digitálních zvuků se vyskytuje problém – jsme omezeni nějakou

maximální hodnotou amplitudy (BÚNO hodnotou 1). Existuje více přístupů, jak dané zvuky smíchat.

Jedno z možných řešení je to, že zvukové stopy jednoduše sečteme, a pokud součet přeteče, uchováme maximální amplitudu. To ovšem neposkytuje adekvátní výsledky, jelikož pokud mixujeme více stop, přetékání je velice časté, což způsobuje častou ztrátu informace signálu.

Jinou možností je výsledek nějak normalizovat – vydělit nějakou hodnotou (např. 2 nebo $\frac{4}{3}$). U hodnoty $\frac{4}{3}$ by pak k přetékání docházelo méně často, u hodnot větších než 2 pak vůbec. Tato metoda však způsobí ztišení výsledného signálu (u normalizace hodnotou 2 by se jednalo o ztišení o $6dB$) oproti realitě. Kdybychom míchali nějaký signál s absolutním tichem, dostali bychom výsledek ztišený. To realitě neodpovídá už vůbec.

Popišme si sofistikovanější metodu. Co bychom vlastně chtěli po výsledném signálu?

Když jeden signál bude absolutní ticho, budeme chtít slyšet druhý signál v plné míře. Uvažme dva signály (A a B) s amplitudami mezi 0 a 1 – chceme, aby výsledná amplituda byla mezi maximální amplitudou a maximem z A a B. To splňuje následující formule:

$$X = A + B - A * B$$

Toto řešení už je celkem uspokojivé, až na to, že platí pouze pro kladné amplitudy.

Pokud pro obě hodnoty menší než 0 upravíme vzorec na $X = A + B + A * B$ a hodnoty rozdílných znamének jednoduše sečteme, dostaneme nejoptimálnější řešení.

Pozn. Pokud chceme smíchat více stop, mixujeme je postupně, proto si uvádíme jen, jak sloučit dvě stopy.

3.7.4 Ozvěna

Algoritmus ozvěny vypadá následovně.

Vezměme si stopu, z níž chceme provádět ozvěnu, a prázdnou stopu. Postupně pro každý vzorek z původní stopy provedeme to, že nezměněnou hodnotu starého streamu smícháme s hodnotou v novém streamu na příslušném místě, a pak v závislosti na zpoždění a útlumu mixujeme s hodnotami na dalších pozicích.

V tomto algoritmu jsme si vzali prázdnou stopu a postupně do ní přidávali (mixovali) posloupnosti hodnot odpovídající jednomu vzorku z původního streamu. Důvod, proč zde uvádím tento „jednoduchý“ algoritmus je ten, abych rozebral zda-li mohu tento efekt použít při simultánním přehrávání a aplikování efektu.

Mohu. Důvod je ten, že algoritmus postupuje chronologicky vzhledem ke zpracování posloupnosti vzorků i aplikaci efektu. To znamená, že mohu přehrát vzorek na pozici, kterou upravuji (samozřejmě až po zpracování vzorku). Doba zpoždění bude přesně taková, jako je doba zpracování jednoho vzorku (tzn. mixování vzorku s výslednou stopou).

3.7.5 Vykreslování

O vykreslování jednotlivých komponent se celkem dobře stará samotné WPF.

Jediný problém, který se vyskytl, je vykreslování samotných stop. Na to, abych vykreslil zvukovou stopu, používám velké množství přímek. Jelikož je to opravdu veliký počet, samotná funkce „DrawingContext.DrawLine“, což je hlavní kreslící funkce přímek ve WPF, je časově velmi náročná.

Namísto toho využívám rychlých kreslících funkcí z assembly „System.Drawing“, kterými kreslím zpracovávanou zvukovou stopu do bitmapy. Ovšem WPF neumí bitmapy vykreslovat jednoduše, proto je nutné bitmapu konvertovat do „BitmapSource“ resp. „ImageSource“. To je sice na celé operaci časově nejnáročnější, avšak i tak je výsledná dosažená rychlost výrazně větší, než kdyby WPF vykreslovalo vše rovnou.

4. Popis řešení

V této kapitole si podrobněji rozepíšeme strukturu aplikace. Jak již bylo řečeno, aplikace se skládá ze dvou hlavních částí (multitrackingové a editační), nad nimiž je jakýsi „správce“. Obrázek viz. 3.1.

4.1 Řídící prvek aplikace

Jedná se o objekt, jehož metoda je volána jako počáteční metoda celé aplikace – má za úkol nastartování programu. Konkrétně se jedná o inicializaci kontextu knihovny OpenAL, vytvoření seznamu použitých stop, vytvoření multitrackingového i editačního okna a nahrání pluginů do seznamu pluginů.

Je to zároveň objekt, jehož metoda se volá jako poslední při ukončení aplikace. Tato metoda řádně likviduje kontext knihovny OpenAL.

„Správce“ má dále za úkol registrovat nové zvukové stopy do svého seznamu registrovaných stop a přepínat dvě hlavní okna (části). Zajišťuje, jaká část má být momentálně aktivní.

4.2 Editační část

Editační část je okno (viz obr. 4.1), které reaguje na konkrétní události. Pro komunikaci s uživatelem využívá různých tlačítek, vstupu z klávesnice a vstupu z myši. Z hlediska GUI je okno složeno z menu, „vykreslovací kontroly“ (tam je vykreslována zvuková stopa), panelu na ovládání přehrávání a dalších tlačítek.

Důležitými prvky objektu okna jsou audio nahrávač, sloužící k nahrávání zvuku, audio přehrávač, sloužící k přehrávání zvuku, objekt nastavení klávesových zkratek (více v samostatné kapitole o zkratkách 4.4), objekt vzhledu (více v samostatné kapitole o designu 4.5) a „kontrola“, která vykresluje aktuální zvukovou stopu – vykreslovací kontrola. Většina událostí zachycených v editačním okně pak volá specifické funkce právě na této kontrole.

(Např. Když okno zachytí událost mazání, tak zavolá funkci mazání na vykreslovací kontrole.)

Rozdělme si reakce na události do několika kategorií:

4.2.1 Akce načítání a ukládání zvukové stopy

Do této sekce patří akce (funkce), reagující na stisk tlačítek resp. klávesových zkratek, týkajících se načítání a ukládání zvukových stop z disku či jiných médií.

Akce **načtení stopy** se pokusí načíst zvukový soubor různých formátů (v práci implementován jen formát Wave PCM). Když se jí to podaří, vytvoří audio přehrávač, přiřadí zvukovou stopu přehrávači a vykreslovací kontrole a také stopu registruje ve správci aplikace. Komunikace s uživatelem (jaký soubor má aplikace nahrát) probíhá přes „OpenFileDialog“.

Akce **uložení stopy** se pokusí uložit stopu do jednoho z možných formátů (v práci implementován jen formát Wave PCM). Komunikace s uživatelem (kam má aplikace soubor uložit) probíhá přes „SaveFileDialog“.

Obrázek 4.1: Editační objekt / okno

Akce **vyčištění stopy** oprostí přehrávač a vykreslovací kontrolu od aktuální stopy. Smaže informační popisky v editačním okně.

4.2.2 Akce editace zvukové stopy

Jedná se o různé stříhání, úpravy hlasitosti a přidávání efektů. Principiálně však fungují téměř stejně.

Editační okno zavolá příslušnou funkci ve vykreslovací kontrole. Ta v závislosti na označení provede danou akci na zvukovou stopu. Tím vznikne nová stopa, která nahradí tu starou (stará stopa však není zahozena, ale je uložena do zásobníku, aby byla možná operace „zpět“).

4.2.3 Akce zobrazování stopy

Tyto akce provádějí pouze kreslící akce ve vykreslovací kontrole. Jedná se o různé přibližování resp. oddalování, posouvání a další různé zobrazování stopy.

Zde se opět setkáme se scénářem, že editační okno zavolá příslušnou funkci ve vykreslovací kontrole, která provede danou akci.

4.2.4 Akce přehrávání a nahrávání

Tato sekce zahrnuje akce přehrávání (spuštění, pozastavení a zastavení) a nahrávání (spuštění, zastavení). Používají funkce z OpenAL.

Akce **spuštění přehrávání** vytvoří nové vlákno, na kterém se provádí vlastní přehrávání díky audio přehrávači a které poskytuje informace o tom, kde se aktuálně přehrávaný stream nachází (pro vypisování času do vykreslovacího okna).

Akce **pozastavení resp. odpauzování přehrávání** uspí resp. vzbudí „přehrávající“ vlákno.

Akce **zastavení přehrávání** zastaví „přehrávající“ vlákno.

Akce **spuštění nahrávání** spustí nahrávání. Pokud ještě není v editačním okně žádná stopa, otevře dialog pro zvolení nahrávacího formátu. Potom spustí na jednom vlákně vlastní nahrávání, které tvoří zvukovou stopu, a když skončí, nahraje ji do editačního okna. Akce spuštění nahrávání ještě spustí vlákno, které informuje o tom, kolik uběhlo času od počátku nahrávání.

Akce **zastavení nahrávání** zapříčiní, aby nahrávací vlákno již nenahrávalo, ale pokračovalo dále ve zpracování právě nahrané stopy.

4.2.5 Akce změny GUI

Jedná se o akce, které většinou změní nějakou vlastnost určité komponenty v editačním okně (Např. zneviditelnění toolbaru pro efekty nebo změna velikosti okna). Tyto akce provádějí pouze triviální akce – např. mění velikosti jiných komponent, mění viditelnosti atd.

4.2.6 Akce změny nastavení

To je funkce, která vytvoří dialog, v němž je možné měnit klávesové zkratky a vzhled. Následně nahradí objekt starých klávesových zkratek za nový, objekt starého vzhledu za nový a spustí transformaci vzhledu.

4.2.7 Akce zachytávání klávesových zkratek

To je reakce na událost uvolnění klávesy. Viz kapitolu 4.4.

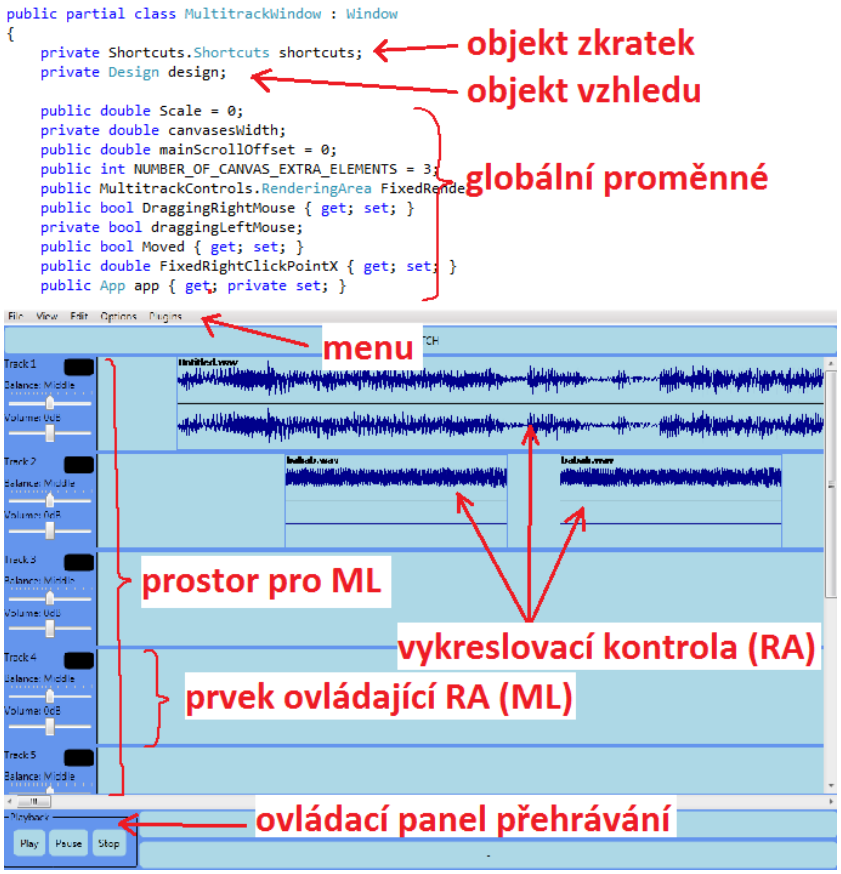
4.3 Multitrackingová část

Multitrackingová část je okno (viz obr. 4.2), které reaguje na konkrétní události. Pro komunikaci s uživatelem využívá různých tlačítek, vstupu z klávesnice a vstupu z myši.

Struktura multitrackingové části je poněkud jiná než struktura editační části. V místě, kde editační okno mělo vykreslovací kontrolu, je prostor pro umístění více ovládacích prvků, které mají za úkol spravovat zvukové stopy. Aplikace není, až na výjimky, řízena událostmi z multitrackingového objektu, ale událostmi z těchto ovládacích prvků. Objekt okna samotného pak slouží pouze ke sjednocení a uchovávání „globálních“ proměnných.

Podobně jako v editační části je vše řízeno událostmi, v okně je umístěn panel, který slouží k ovládání přehrávání, okno obsahuje menu, objekt zkratek a objekt designu.

Vraťme se ještě k prvkům, které ovládají zvukové stopy. Tyto prvky jsou složeny ze dvou částí – levé části, která uchovává nastavení jako například jméno,



Obrázek 4.2: Multitrackingový objekt / okno (MTW)

korekce hlasitosti a vyvážení, a pravé, což je část na vlastní vykreslování konkrétních stop. To se děje velmi podobně jako v editačním okně – stopy ovládají vykreslovací kontroly, které určují, jakým způsobem stopy vykreslovat, pamatují si odkazy na data a prvky, kterým náleží, atd.

Známe již hierarchii od okna až po stopu: multitrackingové okno (MTW) – prvky (ML), ovládající vykreslovací kontroly – vykreslovací kontroly se zvukovou stopou (RA).

Jak již bylo řečeno, aplikace je ovládána hlavně pomocí událostí z ovládacích prvků, které nejsou přímo v multitrackingovém okně, ale jsou trochu více zapouzdřeny. Vyjmenujme si opět nějaké akce.

4.3.1 Načítání a mazání zvukových stop

Obě funkce se váží na událost nějakého konkrétního ovládacího prvku (ML a RA), jež spravuje zvukové stopy.

Načtení vytvoří na dané místo vykreslovací prvek (RA) s odkazy na konkrétní zvukovou stopu, která již je registrována ve správci aplikace. Pokud chce uživatel nahrát stopu „z venku“, aplikace ji normálně nahraje, registruje a až pak s ní ovládací prvek pracuje.

Akce mazání odebere ovládacímu prvku (ML) vykreslovací kontrolu (RA), tudíž danou zvukovou stopu pak nezobrazuje.

4.3.2 Posouvání a označování zvukových stop

Toto jsou reakce na události z prvků jako například posun myši, stisknutí levého či pravého tlačítka myši atd.

Ovládací prvek reaguje na tyto události tím, že vykoná požadovanou funkci – např. posune vykreslovací kontrolu (RA) nebo označí část v ovládacích prvcích (ML).

4.3.3 Zobrazovací funkce

Každá vykreslovací kontrola (RA) v sobě obsahuje informaci o tom, jak se má vykreslovat (měřítko a posunutí). Zobrazovací funkce jsou reakce na události přímo z multitrackingového okna (MTW). Zapříčiní to, že změní informace o vykreslování všem vykreslovacím kontrolám (RA). K nim mají přístup přes ovládací prvky (ML) multitrackingového okna (MTW).

4.3.4 Akce změny nastavení

To je reakce na událost samotného multitrackingového okna (MTW). Viz 4.2.6

4.3.5 Akce zachytávání klávesových zkratek

To je reakce na události samotného multitrackingového okna (MTW). Viz kapitolu 4.4.

4.3.6 Akce přehrávání

To je opět reakce na události samotného multitrackingového okna (MTW) (stisknutí nějakého z tlačítek nebo klávesové zkratky).

V multitrackingové části je přehrávání o dost komplikovanější než v části editační. Složitější je hlavně proto, že aplikace nepřehrává jen jednu stopu, takže nestačí vytvořit jeden audio přehrávač, který streamuje jednu zvukovou stopu. Řešení přehrávání více stop vypadá následovně.

Multitrackingový objekt (MTW) zahrnuje objekt multitrackingového audio přehrávače, který umí přehrávat více stop najednou. Pro každou stopu si vytvoří vlastní zdroj, který přehrává data (buffery) dané stopy. Je to tedy přehrávač, který „paralelně“ přehrává „nabufferovaná“ data z více zdrojů.

Otázka je, jak si příslušná data přehrávač „nabufferuje“. Odpověď je jednoduchá – nijak. přehrávač slouží výhradně k přehrávání a o „bufferování“ se nestará. Od toho je v aplikaci jakýsi nahrávač streamů, který dělá přesně to, že dané streamy „bufferuje“ tam, kam má.

Jak tedy vypadá samotné přehrávání? Aplikace vytvoří vlákno které režíruje přehrávání. Vlákno režie vytvoří jiné vlákno s multitrackingovým audio přehrávačem, který začne přehrávat. Zatím ale nic nepřehrává, poněvadž nemá žádná data „nabufferovaná“. Vlákno režie se postará o to, kdy data načítat. „Nebufferuje“ je přímo, je totiž zaměstnáno samotnou režií, ale vytváří vlákna, kterým přikáže co a kam „nabufferovat“. Tato vlákna načtou data do přehrávače, který je následně přehraje.

4.4 Klávesové zkratky

Tato sekce je věnovaná klávesovým zkratkám. Princip řešení je stejný pro obě hlavní části.

Okno (MTW) obsahuje objekt klávesových zkratek. To je objekt v němž jsou uloženy informace o tom, co má program udělat, když zachytí nějakou zkratku.

Aplikace reaguje na událost uvolnění klávesy. Tehdy se pokusí provést akci, která se váže k dané zkratce. Je nutné podotknout, že aplikace odešle i seznam modifikátorů stisklých ve chvíli uvolnění klávesy. Program tedy reaguje na stisklé modifikátory a uvolněnou klávesu. Když zachytí zkratku, co něco znamená, provede funkci, která je s ní vázaná.

Nastavení klávesových zkratek je principiálně velice jednoduché. Uživatel prostředím GUI vytvoří nový objekt klávesových zkratek svázaných s odpovídajícími funkcemi, který pak nahradí ten starý v aplikaci.

Pozn. modifikační klávesy „Alt“ se nechovají tak, jak by kdokoliv očekával. Levý „Alt“ totiž zároveň znamená systémovou klávesu a pravý znamená modifikaci „Alt“ a „Ctrl“ současně.

4.5 Design

Tato sekce je věnovaná vzhledu aplikace. Princip řešení je stejný pro obě hlavní části.

Objekt okna (MTW) obsahuje objekt designu. Když nějaká funkce přiřadí oknu nový objekt designu, je okno kompletně překresleno. WPF objekty mají určité styly zobrazování. Všechny objekty stejného typu (např. tlačítka, menu, GroupBox atd.) jsou propojeny se svým řídícím objektem, který určuje jejich styl zobrazování. Tomuto propojení se říká „data binding“. Když aplikace mění vzhled objektů jednoho typu, nepředělává styl každého objektu, ale jen toho řídícího. Ostatní se vykreslí podle něho. K změně designu jiných prvků než WPF objektů (např. barva zvukové stopy) používám změnu proměnné, podle které se prvek vykresluje. Např. barva zvukové stopy v multitrackingové části je uložena v proměnné vykreslovací kontroly multitrackingové části. Tato kontrola pak zvukovou stopu vykreslí podle dané proměnné.

4.6 Pluginy

Aplikace (správce aplikace) při spuštění spustí funkci pro registraci pluginů – načte do assembly soubory ze složky „.\Plugins“ s příponou „.dll“. Pokusí se v nich nalézt interface designu či efektu. Když ho nalezne, activator vytvoří instanci objektu a správce si ji uloží do seznamu pluginů. S tímto seznamem pak aplikace pracuje, když se ptá na dostupné pluginy.

5. Programátorská dokumentace

5.1 Vstupní data

Program dokáže načítat soubory z disku přes OpenFileDialog. Soubory musí být ve formátu WAVE PCM, 8-bitové nebo 16-bitové, mono nebo stereo.

Dále program načítá pluginy (soubory s koncovkou „.dll“) ze složky „.\Plugins“. Tyto pluginy by měly obsahovat interface „IEffectInterface“ nebo „IDesignInterface“, aby byly v aplikaci použitelné.

5.2 Závislost tříd

Závislosti a jednotlivých tříd jsou znázorněny na obrázku 5.1.

5.3 Reprezentace dat

Zvuková stopa je reprezentována abstraktní třídou *Track*, která zapouzdřuje různé zvukové formáty.

Třída Track

Od této třídy dědí třídy konkrétních zvukových formátů (v práci je implementována třída *Wave*, která obsluhuje načítání a ukládání formátu WAVE PCM).

Proměnné:

- AudioStream audioStream
  Obsahuje vše, co se zvukových dat týče.
- string Name
  Jméno zvukové stopy, které se užívá v rámci aplikace

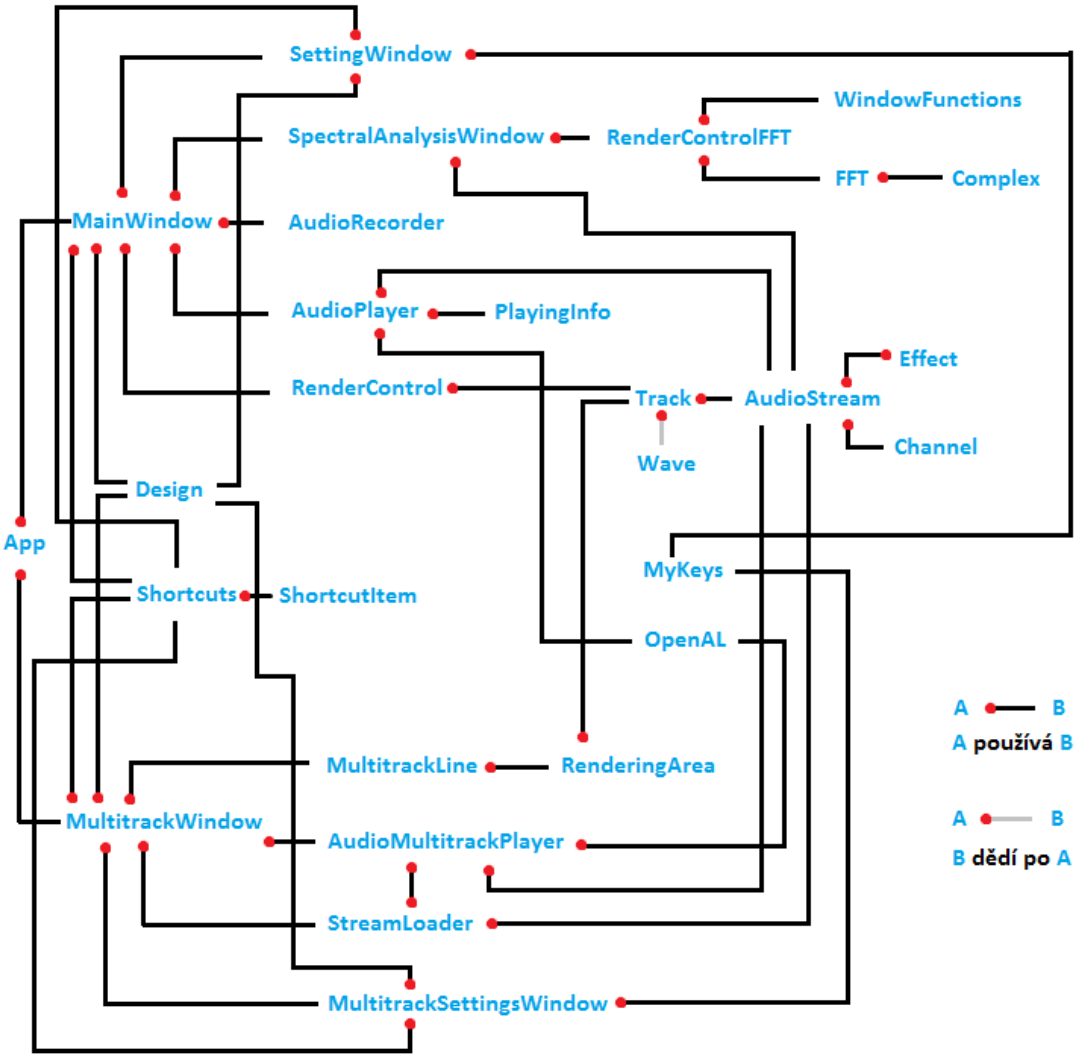
Funkce:

- static Track LoadFile(string name)
  Statická metoda *LoadFile*, nejdříve načte zvukovou stopu z disku (prostřednictvím implementace v odpovídajícím potomkovi), a pak jí registruje ve správci aplikace. Pokud již tam je stopa se stejným jménem, tak právě načtenou stopu přejmenuje.
- abstract void SaveFile(string file)
  Abstraktní metoda pro uložení zvukové stopy.

Třída Wave

Je to potomek třídy *Track*. Obsahuje proměnné odpovídající struktuře souboru WAVE PCM. Viz 2.2. Např. *int chunkSize, short audioFormat.*

Důležité funkce:



Obrázek 5.1: Závislosti tříd ve struktuře aplikace.

- public static new Wave LoadFile(string file)
  Statická metoda, která se pokusí načíst soubor ve formátu WAVE PCM z disku a vrátí objekt *Wave*. Všechna důležitá data jsou uložena v proměnné *AudioStream audioStream*.

- public static Wave CreateWave(byte[] data, byte numChannels, int sampleRate, byte bitsPerSample)
  Statická metoda, jež vytvoří objekt *Wave* z dat v argumentech. Používá se při nahrávání nové stopy.

- public override void SaveFile(string file)
  Uloží obsažená data do souboru na disku ve formátu WAVE PCM.

Třída AudioStream

Nejdůležitější třída, co se zvukových dat týče. Obsahuje vlastní data zvukové stopy, různá další pomocná data pro rychlejší zobrazování, funkce pro počítání indexů, funkce editace atd.

Důležité proměnné:

- const int CONTINUE_POINTS
  Konstanta, která vyjadřuje posloupnost kolika bodů za sebou se bude vykreslovat, aby vykreslení stopy více odpovídalo realitě. Primárně nastaveno na 6.

- int sampleRate
  Vzorkovací frekvence zvukové stopy.

- short bitsPerSample
  Rozlišení zvukové stopy.

- int format
  Číslo formátu. Používá knihovna OpenAL.

- byte[] data
  Vlastní data zvukové stopy. Originální data.

- Channel[] channels
  Indexer, který zprostředkovává získávání konkrétních vzorků z bytového pole.

- byte[][][] DataToView
  Výtažek z originálních dat pro rychlejší zobrazování v editačním okně. Pomocná datová pole.

- double multitrackScale
  Měřítko, určující zobrazování v multitrackingovém okně.

- double[][] DataMultitrack
  Výtažek z originálních dat pro rychlejší zobrazování v multitrackingovém okně. Pomocná multitrackingová pole.

- byte[] CopiedData
  Pole právě kopírovaných dat.

- Stack<byte[]> UndoStack
  Zpětný zásobník. Potřebný k funkci zpět.

- Stack<byte[]> RedoStack
  Dopředný zásobník. Potřebný k funkci dopředu.

- Queue<Effect> effects
  Fronta efektů použitých na stopu.

Důležité vlastnosti:

- double MultitrackScale
  Vrací proměnnou multitrackScale. Při uložení provede přepočítání pomocných dat pro multitrackingový objekt.

- int SampleCount
  Vrací počet vzorků.

- double Duration
  Vrací délku zvukové stopy v sekundách.

- byte[] LastUndoStackData
  Vrací data z vrchu zpětného zásobníku, nebo null.

Důležité funkce:

- int SampleIndex(double second)
  Vrací index vzorků odpovídající dané sekundě.

- int SampleViewIndex(double second, int i)
  Vrací index vzorků odpovídající dané sekundě. *I* znamená, k jakému pomocnému editačnímu poli se index váže.

- public int SampleViewCount(int i)
  Počet vzorků odpovídajícího pomocného editačního pole.

- int PositionInStreamData(double second)
  Vrací index v originálním datovém poli vázaný k dané sekundě. Index odpovídá začátku vzorku.

- AudioStream(short numChannels, int sampleRate, short bitsPerSample, byte[] data)
  Konstruktor objektu. Vytvoří pomocná data a zaplní proměnné.

- void UploadDataMultitrack(double percent)
  Obnoví pomocná multitrackingová data. Ve vstupní proměnné může být hodnota od 0 do 1, což vyjadřuje kolik vzorků oproti originálu bude v pomocných datech. Je volána při změně dat a při změně *Scale*.

- void UploadViewData()
  Vytvoří pomocná editační data. Funkce je volána při změně dat.

- int LoadStream(byte[] buffer, int from, int count, int maxTo)
  Načte originální data do konkrétního bufferu.

- byte[] Copy(double begin, double end)
  Zkopíruje originální data v závislosti na začátku a konci zvukové stopy uváděných v sekundách. Vytvoří pole, jež uloží do proměnné *CopiedData*.

- void Paste(double where)
  Vloží *CopiedData* do originálních dat na pozici odpovídající dané sekundě. Stará data uloží na zásobník. Provede upload pomocných dat.

- void Delete(double begin, double end)
  Vymaže originální data v závislosti na začátku a konci zvukové stopy uváděných v sekundách. Stará data uloží na zásobník. Provede upload pomocných dat.

- void Undo()
  Data z vrchu zpětného zásobníku posune do originálních dat. Originální data posune do dopředného zásobníku. Provede upload pomocných dat.

- void Redo()
  Data z vrchu dopředného zásobníku posune do originálních dat. Originální data posune do zpětného zásobníku. Provede upload pomocných dat.

- void Upload(byte[] newData)
  Nová data uloží namísto originálních dat. Originální data posune na zpětný zásobník. Vymaže dopředný zásobník. Provede upload pomocných dat.

- void AddEffect(Effect effect)
  Přidá objekt efektu do fronty (*Queue<Effect> effects*).

- void DoEffects()
  Postupně aplikuje efekty z fronty na data. Originální data uloží na zpětný zásobník. Provede upload pomocných dat.

Třída Channel

Jelikož data zvuku jsou uložena jako posloupnost bytů, slouží tato třída jako indexer pro pohodlnější používání.

Proměnné:

- short numChannels
  Udává, kolik kanálů má zvuková stopa.

- short bitsPerSample
  Rozlišení zvukové stopy.

- int number
  Vlastní číslo kanálu.

- byte[] data
  Odkaz na data.

Funkce:

- unsafe double At(int i)
  Vrací hodnotu vzorku odpovídající kanálu na pozici *i*. Vrací hodnoty od 0 do 1.

- unsafe void To(int i, double value)
  Obdoba funkce *At* až na to, že hodnotu ukládá.

- unsafe Int64 this[int i]
  Vrací, nebo ukládá hodnotu vzorku odpovídajícího kanálu na pozici *i*. Typ hodnot odpovídá rozlišení. Např. 8-bitové rozlišení pracuje s hodnotami 0-255 (byte).

5.4 Správce aplikace

Objekt třídy *App*. Nejširší objekt zapouzdřující celou aplikaci. Hlavní řídící prvek, který spravuje pluginy, registraci zvukových stop a obě okna.

Třída App

Tato třída dědí od třídy *Application*.

Proměnné:

- MainWindow mw
  Objekt reprezentující editační okno.

- MultitrackWindow mltw
  Objekt reprezentující multitrackingový objekt.

- List<Track> UsedTracks
  Seznam registrovaných zvukových stop.

- Dictionary<string, Plugins.Interfaces.IEffectInterface> EffectPlugins
  Seznam efektových pluginů.

- Dictionary<string, Plugins.Interfaces.IDesignInterface> DesignPlugins
  Seznam designových pluginů.

- bool isMainWindowOn
  Proměnná, která vyjadřuje, jaké okno je právě aktivní. Pokud je pravdivá, je aktivní editační okno, jinak multitrackingové.

Funkce:

- void Application_Startup(object sender, StartupEventArgs e)
  Obsluha události začátku aplikace. Vytváří globální kontext OpenAL. Vytvoří obě okna. Načte pluginy.

- void Switch()
  Prohodí okna.

- void RegisterTrack(Track track)
  Přidá zvukovou stopu do seznamu registrovaných stop a pojmenuje ji.

- Application_Exit(object sender, ExitEventArgs e)
  Obsluha události konce aplikace. Ruší globální kontext OpenAL.

- void LoadPlugins()
  Načte pluginy (soubory s koncovkou „.dll“) ze složky „.\Plugins“ do svých seznamů.

5.5 Editační část

Editační část je reprezentována objektem *MainWindow mw* ze správce aplikace. V GUI je to okno s ovládacími prvky, ve kterém se zobrazuje zvuková stopa.

Třída MainWindow

Potomek třídy *Window*.

Důležité „Xaml“ objekty:

- RenderControl renderControl1
  Nejdůležitější prvek editační části. Do něj je vykreslována editovaná stopa. Přes něj probíhají zobrazovací a editační funkce.

- Button ButtonStyler
  Tlačítko určující styl ostatních tlačítek – přes databinding.

- GroupBox GroupBoxStyler
  Groupbox určující styl ostatních groupboxů – přes databinding.

- Rectangle rectSelection
  Označení ve stopě.

- Line lineMainLine
  Přímka označující „nakliklé“ místo ve stopě.

- Line linePlayingLine
  Přímka označující právě přehrávané místo.

Proměnné:

- AudioRecorder recorder
  Audionahrávač.

- AudioPlayer player
  Audiopřehrávač.

- System.Threading.Thread playingThread
  Vlákno používané k přehrávání.

- Shortcuts.Shortcuts shortcuts
  Objekt klávesových zkratek.

- App app
  Odkaz na správce aplikace.

- Design Design
  Objekt vzhledu. Při přiřazení volá funkci *DoDesign(value)*.

Důležité funkce. Do této sekce patří i obsluhy událostí – ty však většinou spustí odpovídající funkce bez parametrů, proto takovéto obsluhy nebudu uvádět. Např. funkce *btnLoad_Click(object sender, RoutedEventArgs e)* spustí odpovídající funkci *Load()*. Tato struktura je zvolená proto, aby bylo možné provádět klávesové zkratky jednoduše.

- Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
  Obsuha události zavření okna. Zruší objekt audiopřehrávače (uklízení po OpenAL) a vyšle signál, aby se zavřela celá aplikace.

- void Load()
  Vytvoří *OpenFileDialog* k nahrání souboru. Načte soubor (vytvoří objekt *Track*) a přiřadí ho do *player* a *renderControl1*.

- void Save()
  Uloží zvukovou stopu z *renderControl1* přes *SaveFileDialog* na disk.

- void Clear()
  Vyčistí proměnné, aby nebyly vázány na žádnou zvukovou stopu.

- void Play()
  Spustí přehrávání v objektu *player* na vlákně *playingThread*.

- void Pause()
  Pozastaví, nebo odpauzuje přehrávací vlákno. Pozastaví, nebo odpauzuje nahrávání (nastaví proměnnou *recorder.Paused*).

- void Stop()
  Zastaví přehrávací vlákno. Zastaví nahrávání (nastaví proměnnou *recorder.Active* na „false“).

- void Record()
  Spustí na jiném vlákně nahrávání v objektu *recorder*. Poté, co přestane nahrávat (*recorder.Active* někdo změní na „false“), registruje nahranou stopu a přiřadí ji do *player* a *renderControl1*. Pozn. Pokud je nahrávání aktivní, tak ho ukončí a nic jiného nedělá.

- void RecordOnThread()
  Nahrávací metoda. Je pouštěna funkcí *Record()* na samostatném vlákně.

- void Undo()
  Provede akci *Undo()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Redo()
  Provede akci *Redo()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Delete()
  Provede akci *Delete()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Copy()
  Provede akci *Copy()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Paste()
  Provede akci *Paste()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Cut()
  Provede akci *Copy()* a *Delete()* na streamu stopy v *renderControl1* v závislosti na označení.

- void Crop()
  Zkopíruje část stopy v závislosti na označení a zavolá *Upload(byte[] newData)* na streamu stopy v *renderControl1*.

- void DoSpectralAnalysis()
  Vytvoří dialog spektrální analýzy v závislosti na aktuální zvukové stopě, pozici *mainLine* a designu.

- Window_KeyUp(object sender, KeyEventArgs e)
  Provede akci příslušné klávesové zkratky. Zavolá *shortcuts.DoShortcut(e, Keyboard.Modifiers)*.

- void DoDesign(Design design)
  Změní design. Globální předělání vzhledu řeší tak, že změní vzhled pouze nabindovaným objektům a ostatní se přizpůsobí.

- void Settings_Click()
  Vytvoří dialog nastavení, který mění design editační části (nastavuje proměnnou *Design*), a vytváří klávesové zkratky. Poté je přiřadí do proměnné *shortcuts*. Při „stornu“ nahraje původní design a zkratky nepřiřazuje.

- Funkce provádějící změnu zobrazení, např. *void Zoom_in()*, *void Zoom_selection()* či *void Zoom_out_vertically()*.
  Zavolají příslušné funkce na *renderControl1*, která stopu zobrazuje.

- Funkce provádějící efekty, např. *void DoPluginEffect(object sender, RoutedEventArgs e)* nebo *void btnDelay(object sender, RoutedEventArgs e)* či *VolumeChangeUp()*.
  Provedou příslušné efekty na streamu stopy v *renderControl1* v závislosti na označení. Je to posloupnost metod *renderControl1.Track.AudioStream.AddEffect(Effect effect)* a *renderControl1.Track.AudioStream.DoEffects()*.

Třída RenderControl

Třída, která vykresluje zvukovou stopu. Obsahuje funkce obsluhující zobrazování a editaci zvukové stopy. Tyto funkce volá editační okno (rodič).

Hlavní proměnné:

- Track Track
  Odkaz na zobrazovanou zvukovou stopu. Při přiřazení se provede překreslení.
- MainWindow mw
  Odkaz na editační okno.
- double trackFrom
  Počátek zobrazované části zvukové stopy v sekundách.
- double trackTo
  Konec zobrazované části zvukové stopy v sekundách.
- double zoom
  Vertikální zoom. Hodnota 1 je identické zobrazení, větší než 1 je roztáhnutí, menší je zmenšení.
- double selectionFixedPoint
  Pozice ukotvení při kliknutí myši.

Hlavní funkce:

- Funkce provádějící změnu zobrazení, např. *Zoom_ in()*, *Zoom_ selection()* či *Zoom_ out_ vertically()*.
  Provedou úpravu proměnných *trackFrom*, *trackTo* a *zoom*. Poté zavolají vykreslení.
- Funkce provádějící editaci (*Delete()*,*Copy()*,*Crop()*,*Cut()*, *Paste()*).
  Princip popsán již ve třídě *MainWindow*.
- Funkce provádějící efekty, např. *VolumeChange(string UID)* nebo *FadeUp-Down(string UID)*, *DoDelay(string UID, object[] parameters)* či *DoEffect(WaveEditor.Plugins.Interfaces.IEffectInterface IEffect)*.
  Provedou příslušné efekty na streamu stopy v závislosti na označení. Je to posloupnost metod *Track.AudioStream.AddEffect(Effect effect)* a *Track.AudioStream.DoEffects()*.
- void InicializeBeginPlaying()
  Připraví zobrazovací přímky na přehrávání. Upraví jejich viditelnosti přes *Dispatcher*.
- void InicializeStopPlaying()
  Připraví zobrazovací přímky na konec přehrávání. Upraví jejich viditelnosti přes *Dispatcher*.
- void ChangePlayingLinePosition(double x)
  Přes *Dispatcher* posouvá přehrávací přímku na danou pozici v *RenderControl*.

- int ChooseIndexOfShownData()
  Vrátí index pole pomocných dat pro zobrazení v závislosti na šířce zobrazení.

- override void OnRender(DrawingContext drawingContext)
  Vlastní metoda vykreslování. Vykreslí danou zvukovou stopu, pozadí a dělící čáry.

- double GetPositionInTrack(double x)
  Vrátí pozici ve zvukové stopě v sekundách. *X* je x-ová pozice kurzoru v příslušné *RenderControl*.

- void DoRecordWatches(object audioRecorder)
  Funkce k aktualizování zobrazovaného času při nahrávání. Musí se volat na samostatném vlákně. Je závislá na objektu *audioRecorder*. Jakmile *audioRecorder* skončí nahrávání, skončí i samotná funkce.

5.5.1 Přehrávací a nahrávací třídy

Třída AudioPlayer

Třída potřebná k přehrávání zvukové stopy. K přehrávání používá funkce OpenAL. Potomek třídy „IDisposable“.

Hlavní proměnné:

- AudioStream audioStream
  Odkaz na stream zvukové stopy.

- int[] FBuffers
  ID bufferů používaných ve funkcích OpenAL.

- int FSource
  ID zdroje tohoto přehrávače.

- byte[] buf
  Buffer potřebný k nahrávání dat do bufferů OpenAL.

- PlayingInfo playingInfo
  Informace o přehrávání.

Hlavní funkce:

- AudioPlayer(AudioStream audioStream)
  Konstruktor. Inicializuje buffery, registruje zdroj a nastaví parametry pro OpenAL.

- bool Playback()
  Nahraje první buffery při puštění přehrávače a spustí přehrávání v OpenAL. Pokud už nemá co bufferovat, vrátí „false“, jinak „true“.

- bool Stream(int buffer)
  Načte data do příslušného bufferu. Pokud už nemá co bufferovat, vrátí „false“, jinak „true“.

- bool Update()
  Načte data do všech volných bufferů. Pokud už nemá co bufferovat, vrátí „false“, jinak „true“.

- void Empty()
  Uvolní všechny buffery.

- void Play(object playingInfo)
  Přehrává zvukovou stopu podle příslušných vstupních parametrů. Je nutné pustit na novém vlákně.

- void Stop()
  Zastaví přehrávání a uvolní buffery.

- void Pause()
  Pozastaví přehrávání zdroje.

- void UnPause()
  Pokračuje v přehrávání zdroje.

- void Dispose()
  Zastaví přehrávání. Uvolní buffery a zdroj.

Třída PlayingInfo

Třída využívaná v přehrávání. Jsou v ní uložené informace o úseku přehrávání a nabufferované pozici.

Hlavní proměnné:

- int StreamPosition
  Pozice posledního nabufferovaného vzorku ve streamu.

- int End
  Index konce ve streamu.

- int Begin
  Index začátku ve streamu.

- RenderControl renderControl
  Odkaz na *RenderControl*, v níž posouvá přehrávací vlákno přehrávací přímku.

Hlavní funkce:

- void ShowPlayedMainLine()
  Nastaví přehrávací přímku v příslušné *RenderControl* na danou pozici.

- double GetStreamPositionPercents()
  Procentuální pozice v příslušné *RenderControl*.

Třída AudioRecorder

Třída využívaná k nahrávání.

Hlavní proměnné:

- int SampleRate
  Vzorkovací frekvence.
- byte NumChannels
  Počet kanálů.
- byte BitsPerSample
  Rozlišení.
- int format
  Číslo formátu. Závisí na výše jmenovaných proměnných. Potřebné pro OpenAL.
- byte[] myBuffer
  Buffer využívaný mezi OpanAL a aplikací při nahrávání.
- List<byte[]> data
  Seznam nahraných úseků. Chronologicky seřazená pole nahraných dat.
- bool Active
  Značí, jestli má rekordér ještě nahrávat, nebo ne.
- bool Paused
  Indikátor pauzy.

Hlavní funkce:

- void SetValues(int sampleRate, byte numChannels, byte bitsPerSample)
  Nastaví proměnné objektu podle vstupních parametrů.
- byte[] StartRecording()
  Spustí nahrávání. Skončí, když je *Active* „false“. Vrátí nahraná data.
- byte[] CreateDataArray(byte[] data, int samples)
  Vytvoří pole bytů v závislosti na parametrech – bufferu a počtu vzorků v něm.
- byte[] dataConcat()
  Spojí list polí *data* do jediného pole, které vrátí.

5.5.2 FFT třídy

Zde uvedu třídy potřebné k Fourierově transformaci a vytvoření spektrogramu.

Třída Complex

Třída reprezentující komplexní čísla (včetně operací).

Proměnné:

- double real
  Reálná část komplexního čísla.

- double imag
  Imaginární část komplexního čísla.

Funkce: V této třídě jsou definovány všechny elementární funkce komplexních čísel operátory. Pojmenování operátorů, arity operací i význam operací je intuitivní. Operace jsou následující: „+, -, *, /“.

Třída FFT

Statická třída definující funkce týkající se FFT.

Obsažené funkce:

- static Complex[] DoFFTForwards(double[] input)
  Vykoná transformaci na „polynom“ a vrátí transformovaný „polynom“.

- static Complex[] DoFFTBackwards(Complex[] input)
  Provede zpětnou transformaci.

- static Complex[] transform(Complex[] pol, int sqr, Complex[] omegas)
  Rekurzivní funkce Fourierovy transformace. *Pol* je vstupní polynom, *sqr* je stupeň transformace a *omegas* je pole předpočítaných „odmocnin z jedné“.

- static Complex count_k_primitive_sqr(int k)
  Spočte primitivní k-tou odmocninu z jedné.

- static int Log2(int n)
  Rychlá implementace dvojkového logaritmu.

- static bool IsPowerOfTwo(int n)
  Rychlá funkce pro zjištění, zda-li je vstup mocninou dvojky.

- double[] DoSpectrogram(double[] input)
  Vrátí spektogram (posloupnost hodnot) daného vstupu. Využívá FFT. Spektrogram ovšem není úplný, je ve tvaru FFT – dvojnásobný (sudý pro amplitudu, lichý pro fázi) a posunutý o jedna vpravo.

- double[] DoSpectrogram(Complex[] trandsformedPolynom)
  Vrátí spektrogram (posloupnost hodnot) transformovaného vstupu. Spektrogram ovšem není úplný, je ve tvaru FFT – dvojnásobný (sudý pro amplitudu, lichý pro fázi) a posunutý o jedna vpravo.

Třída WindowFunctions

Statická třída definující „Window“ funkce. Funkce vždy vrací posloupnost hodnot od 0 do 1, která je dlouhá podle vstupního parametru. Např. funkce *double[] Rectangular(int length)* vytvoří posloupnost jedniček velikosti *length*.

Definované funkce jsou: Rectangular, Hann, Hamming, TriangularZeroEnds, TriangularNonZeroEnds, Sine, BartlettHann a BlackmanHarris.

Třída SpectralAnalysisWindow

Třída, jež dědí z *Window*. Objekt *SpectralAnalysisWindow* je volán jako dialog. Slouží k spektrální analýze.

Hlavní proměnné:

- delegate double[] WindowFunction(int length)
  Delegát funkce, jež reprezentuje „Window“ funkci.
- List<WindowFunction> windowFunctions
  Seznam dostupných „Window“ funkcí.
- WindowFunction currentWindowFunction
  Používaná „Window“ funkce.
- AudioStream stream
  Odkaz na stream.
- int Position
  Pozice ve streamu. Při změně se vše přepočítá a zobrazí.
- RenderControlFFT renderControl
  Vykreslovací kontrola používaná k vykreslení spektrogramu. Obsahuje zobrazovací funkce.

Hlavní funkce:

- void RedrawRenderControl()
  Vytvoří spektrogramy (pro více kanálů se vytvoří více spektrogramů) z daného streamu a pozice v něm a přiřadí je vykreslovací kontrole.
- void Export_Click(object sender, RoutedEventArgs e)
  Uložit obrázek z vykreslovací kontroly na disk. Používá k tomu *SaveFileDialog*.

Třída RenderControlFFT

Vykreslovací kontrola využívaná v *SpectralAnalysisWindow*. Obsahuje informace o tom, jak se má spektrogram vykreslit a podle toho pracuje. Dovoluje uživateli měnit zobrazení přes události, které mění informace o zobrazování.

Hlavní proměnná:

- double[][] Values
  Hodnoty spektrogramu. Při dosazení se kontrola překreslí.

Hlavní funkce:

- override void OnRender(System.Windows.Media.DrawingContext drawingContext)
  Vykreslí spektrogram na své pozadí.

5.6 Multitrackingová část

Editační část je reprezentována objektem *MultitrackWindow mtw* ze správce aplikace. V GUI je to okno s ovládacími prvky, ve kterém se zobrazují zvukové stopy. Poskytuje zobrazování více stop na úkor editace.

Třída MultitrackWindow

Potomek třídy *Window*.

Důležité „Xaml" objekty:

- Button ButtonStyler
  Tlačítko určující styl ostatních tlačítek – přes databinding.
- GroupBox GroupBoxStyler
  Groupbox určující styl ostatních groupboxů – přes databinding.
- StackPanel MultitrackStackPanel
  StackPanel ve kterém leží prvky typu *MultitrackLine*, které zobrazují zvukové stopy.

Hlavní proměnné:

- double Scale
  Měřítko, které zajišťuje, aby se stopy vykreslovaly v odpovídajícím poměru. Horizontální zoom.
- RenderingArea FixedRenderingArea
  Odkaz na přenášený prvek vykreslující zvukovou stopu.
- Shortcuts.Shortcuts shortcuts
  Objekt klávesových zkratek.
- Design Design
  Objekt vzhledu. Při přiřazení volá funkci *DoDesign(value)*.
- Thread playingThread
  Vlákno, které řídí přehrávání.

Hlavní funkce:

- void ResetVariables()
  Zruší nastavení úchytů přímek a označení.
- void CompleteRedraw()
  Překreslí všechny komponenty.

- void DoDesign(Design design)
  Změní vzhled okna. Globální předělání vzhledu řeší tak, že změní vzhled pouze nabindovaným objektům a ostatní objekty se přizpůsobí.

- void ShiftTracksInCanvas(double value)
  Posune vykreslovací komponenty o danou hodnotu vzhledem k připoutání vykreslovací komponenty.

- void Mainscroll_Scroll(object sender, System.Windows.Controls.Primitives.-ScrollEventArgs e)
  Správce události posunutí horizontálního scrollbaru. Posune vykreslovací kontroly.

- void UpdateMainscroll()
  Updatuje velikost horizontálního scrollbaru. Funkce se volá při změně posunutí nebo velikostí prvků.

- bool MultitrackIsEmpty
  Vrací „true“, pokud není zobrazováva žádná stopa.

- bool MouseCursorIsInMultitrack()
  Vrací „true“, pokud je kursor myši v prostředí, kam se mohou vykreslovat zvukové stopy.

- void Play()
  Spustí na *playingThread* správce přehrávání, který vytvoří multitrack audio přehrávač a vytváří bufferovací vlákna.

- object[] GetChildren()
  Přes *Dispatcher* vrátí potomky objektu *MultitrackStackPanel*.

- int RACout()
  Vrátí počet vykreslovacích kontrol.

- void StartPlay(object info)
  Správce přehrávání. Vytvoří multitrack audio přehrávač a vytváří bufferovací vlákna.

- void btnPause_Click(object sender, RoutedEventArgs e)
  Pozastaví, nebo odpauzuje *playingThread*.

- void Stop()
  Zastaví přehrávání.

- void Clear(object sender, RoutedEventArgs e)
  Vyčistí zobrazovací prostor.

- void ZoomIn(object sender, RoutedEventArgs e)
  Dvojnásobně přiblíží zvukové stopy.

- void ZoomOut(object sender, RoutedEventArgs e)
  Dvojnásobně oddálí zvukové stopy.

- void Settings_Click()
  Vytvoří dialog nastavení, který mění design multitrackingové části (nastavuje proměnnou *Design*) a vytváří klávesové zkratky. Poté je přiřadí do proměnné *shortcuts*. Při „stornu“ nahraje původní design a zkratky nepřiřazuje.

- void Window_KeyUp(object sender, KeyEventArgs e)
  Provede akci příslušné klávesové zkratky. Zavolá *shortcuts.DoShortcut(e, Keyboard.Modifiers)*.

Třída MultitrackControls.MultitrackLine

Potomek *UserControl*. Třída, která je v designu oddělená na levou (nastavovací část) a pravou (zobrazovací část), kam jsou vykreslovány zvukové stopy. Umožňuje načítat stopy z disku nebo vkládat již registrované stopy. Stopy jsou vykreslovány do pravé části do vykreslovacích kontrol.

Důležité „Xaml“ objekty:

- Button ButtonStyler
  Tlačítko určující styl ostatních tlačítek – přes databinding.

- GroupBox GroupBoxStyler
  Groupbox určující styl ostatních groupboxů – přes databinding.

- Canvas renderingCanvas
  V tomto objektu jsou ukotvovány vykreslovací komponenty.

Hlavní proměnné:

- MultitrackWindow mtw
  Odkaz na rodičovské multitrackingové okno. Slouží jako koordinátor jednotlivých *MultitrackLine*.

- List<RenderingArea> trackList
  Seznam vykreslovacích komponent v *renderingCanvas*.

- Design design
  Odkaz na design.

Hlavní funkce:

- void InsertFromFile(object sender, RoutedEventArgs e)
  Načte zvukovou stopu z disku, registruje ji ve správci aplikace, vytvoří vykreslovací komponentu, kterou přiřadí do *renderingCanvas*, a pak ji vykreslí.

- void InsertExistTrack(object sender, RoutedEventArgs e)
  Vytvoří vykreslovací komponentu, kterou přiřadí do *renderingCanvas*, a pak ji vykreslí.

- void UserControl_PreviewMouseMove(object sender, MouseEventArgs e)
  Pokud uživatel drží stisklé levé tlačítko myši, označuje úsek v *renderingCanvas*. Pokud drží stisklé tlačítko myši pravé, posouvá vykreslovací kontrolu (tu, na kterou uživatel klikl první).

- void renderingCanvas_ContextMenuOpening(object sender, ContextMenuEventArgs e)
  Vytvoří kontextové menu *renderingCanvas*.

- void renderingCanvas_MouseEnter(object sender, MouseEventArgs e)
  Pokud přetahuje uživatel nějakou vykreslovací kontrolu z jiného objektu *MultitrackLine*, tak objekt vykreslovací kontroly přemístí do aktuální *MultitrackLine*.

- void Thumb_DragDelta(object sender, System.Windows.Controls.Primitives.DragDeltaEventArgs e)
  Vertikálně rozšiřuje*MultitrackLine*.

- void ExpandHeader(object sender, RoutedEventArgs e)
  Expanduje, nebo schová část ovládacích prvků z hlavičky.

- void DoDesign(Design design)
  Změní design. Globální předělání vzhledu řeší tak, že změní vzhled pouze nabindovaným objektům a ostatní se přizpůsobí.

Třída MultitrackControls.RenderingArea

Potomek *UserControl*. Vykreslovací kontrola. Vykresluje zvukovou stopu.

Hlavní proměnné:

- Track track
  Odkaz na vykreslovanou zvukovou stopu.

- MultitrackLine mtl
  Odkaz na rodičovský objekt *MultitrackLine*.

- int Index
  Index v seznamu vykreslovacích kontrol rodičovského objektu *MultitrackLine*.

- double LeftCanvas
  Pozice v *renderingCanvas* rodičovského objektu *MultitrackLine*.

- Design design
  Objekt vzhledu.

Hlavní funkce:

- double RightCanvas
  Vrací pozici pravé části kontroly v *renderingCanvas* rodičovského objektu *MultitrackLine*.

- override void OnRender(DrawingContext drawingContext)
  Vykreslí na pozadí zvukovou stopu.

- void UserControl_MouseDoubleClick(object sender, MouseButtonEventArgs e)
  Přepne do editačního okna, kde zobrazí danou zvukovou stopu.

- void Delete(object sender, RoutedEventArgs e)
  Odstraní kontrolu z rodičovského objektu *MultitrackLine*.

5.6.1 Přehrávací třídy

Třída MultitrackControls.AudioMultitrackPlayer

Potomek třídy *IDisposable*. Přehrává nabufferovaná data z více zdrojů.

Hlavní proměnné:

- int[][] FBuffers
  Pole bufferů odpovídajících zdrojů.
- Queue<int>[] waitingBuffersArray
  Pole front čekajících bufferů (na přehrání). Jednotlivé fronty odpovídají zdrojům.
- Queue<int>[] freeBuffersArray
  Pole front volných bufferů (mohou se použít k bufferování). Jednotlivé fronty odpovídají zdrojům.
- int[] FSources
  Pole zdrojů.

Hlavní funkce:

- bool Playing(int index)
  Zjistí, jestli hraje příslušný zdroj.
- bool Update(int index)
  Nahraje čekající buffery do OpenAL, kde je přehraje a uvolní je. Vrací „true“, pokud nahrál nejaký buffer do OpenAL.
- void EmptyAll()
  Uvolní všechny buffery.
- void Empty(int index)
  Uvolní buffery daného zdroje.
- void Play()
  Pokouší se přehrávat zdroje. Když má něco nabufferováno, tak to přehraje, jinak mlčí. Upozornění – je to nekonečná smyčka, musí se sestřelit.
- void Stop()
  Zastaví přehrávání všech zdrojů přes OpenAL.
- void Pause()
  Pozastaví přehrávání všech zdrojů přes OpenAL.
- void UnPause()
  Odpauzuje přehrávání všech zdrojů přes OpenAL.
- void Dispose()
  Uvolní všechny buffery a zdroje OpenAL.

Třída MultitrackControls.StreamLoader

Třída potřebná na načítání streamů do *AudioMultitrackPlayer*.

Hlavní proměnné:

- AudioStream audioStream
  Odkaz na stream.
- Queue<int> waitingBuffers
  Odkaz na čekající buffery konkrétního zdroje.
- Queue<int> freeBuffers
  Odkaz na volné buffery konkrétního zdroje.
- int FSource
  Konkrétní zdroj.

Funkce:

- void PrepareBuffer(object param)
  Načítá buffery ze streamu do multitrack audio přehrávače. Odkud a kam má načítat pozná ze vstupních parametrů, což jsou indexy začátečního a koncového vzorku (*int[2]*).
- bool Load(int buffer, ref int from, int to)
  Načte data ze streamu do příslušného bufferu OpenAL. Posune proměnnou *from* o načtená data kupředu.

5.7 Efekty

Všechny efekty jsou potomky třídy *Effect*.

Třída Effect

Abstraktní třída pro efekty. Potomci si definují vlastní tělo metody *DoEffect()* a mohou obsahovat i jiné objekty. S tímto rozhraním umí pracovat stream zvukové stopy.

Struktura:

- AudioStream stream
  Odkaz na příslušný stream.
- double from_
  Začátek efektu v sekundách.
- double to_
  Konec efektu v sekundách.
- abstract void DoEffect()
  Funkce, která mění aktuální stream. Každý potomek si ho musí implementovat sám.

5.8 Design

Třída, přes kterou aplikace nastavuje svůj vzhled.

Třída Design

Obsahuje pouze definice proměnných. Všechny designové objekty v aplikaci jsou třídy, které jsou potomky třídy *Design*. Tyto třídy ve svém konstruktoru doplní objekty do prázdných proměnných rodiče (*Design*). Potomci této třídy nedefinují nic jiného.

Proměnné:

- Pen RenderCentrol_penTrack
  Barva vykreslované stopy.
- Pen RenderCentrol_penZeroChannelLine
  Barva přímky středu zvukové stopy.
- Pen RenderCentrol_penInterChannelLine
  Barva oddělujících čar mezi kanály a vykreslovacími komponentami.
- Brush RenderCentrol_background
  Pozadí vykreslovacích kontrol.
- Brush TimeBackground
  Pozadí časového rámečku a hlaviček.
- Line RenderControl_MainLine
  Barva a tloušťka kotvící přímky.
- Rectangle RenderControl_Selection
  Barva a průhlednost označení.
- Line RenderControl_PlayingLine
  Barva a tloušťka přehrávající přímky.
- Brush BorderBrush
  Barva rámečků.
- Brush LabelTextColor
  Barva textu času.
- Brush TextColor
  Barva názvu stopy.
- Brush WindowBackground
  Pozadí hlavních oken.
- Brush RenderControl_InfoNormalLabelBackground
  Barva pozadí normálních popisků v *MainWindow*.
- Brush RenderControl_InfoHeaderLabelBackground
  Barva pozadí hlavičkových popisků v *MainWindow*.

- Brush RenderControl_InfoNormalLabelForeground
  Barva popředí normálních popisků v *MainWindow*.

- Brush RenderControl_InfoHeaderLabelForeground
  Barva popředí hlavičkových popisků v *MainWindow*.

- Thickness GroupBoxStylerBorderThicknes
  Šířka ohraničení objektů typu *GroupBox*.

- Brush GroupBoxStylerBorderBrush
  Barva ohraničení objektů typu *GroupBox*.

- Style ButtonStyle
  Styl tlačítek.

5.9 Zkratky

Třída Shortcuts

Hlavní třída týkající se zkratek.

Proměnné:

- List<ShortcutItem> shortcutsMap
  Seznam používaných zkratek s odpovídajícími funkcemi.

- MainWindow mainWindow
  Odkaz na editační okno.

- MultitrackWindow mtw
  Odkaz na multitrackingové okno.

Funkce:

- Shortcuts(MainWindow mw)
  Konstruktor, který vytvoří základní *shortcutsMap* a prováže ho s *mainWindow*.

- Shortcuts(MultitrackWindow mtw)
  Konstruktor, který vytvoří základní *shortcutsMap* a prováže ho s *this.mtw*.

- Shortcuts Copy()
  Vytvoří duplikát objektu.

- void DoShortcut(KeyEventArgs e, ModifierKeys mk)
  Pokud existuje klávesová zkratka jako je v parametrech, provede funkci s ní svázanou.

Třída ShortcutItem

Třída záznamu klávesové zkratky.

Proměnné:

- delegate void RepresentedFunction()
  Definice delegáta funkce svázané se zkratkou.
- string Name
  Jméno, pod kterým daná zkratka vystupuje.
- bool Exist
  Existence dané zkratky.
- ModifierKeys Mk
  Modifikační klávesy ve zkratce.
- Key K
  Nemodifikační klávesa ve zkratce.
- int Uid
  Pozice zkratky v *shortcutsMap*. Důležité pro nastavování a vypisování do *Label*.
- RepresentedFunction rp
  Funkce svázaná se zkratkou.
- System.Windows.Controls.Label Label
  Objekt *Label* svázaný se zkratkou. Skrz něj komunikuje s uživatelem při nastavování.

Funkce:

- ShortcutItem(string name, ModifierKeys mk, Key k, RepresentedFunction rp, bool exist, int uid)
  Konstruktor.
- ShortcutItem Copy()
  Funkce, která duplikuje objekt klávesové zkratky.
- void DoShortcutFunction()
  Provede funkci svázanou s danou klávesovou zkratkou.

Třída MyKeys

Třída potřebná k nastavování zkratek.

Hlavní proměnné:

- bool waitingForAKey
  Indikuje, jestli nastavuje uživatel nějakou zkratku.
- ModifierKeys modKey
  Stisklé modifikační klávesy.

- System.Windows.Controls.Label Label
  Svázání nastavované klávesové zkratky s odpovídající *Label*. Při přiřazení nastaví *waitingForAKey* na „true“, jelikož se přiřazení děje právě při začátku nastavování konkrétní zkratky.

Funkce:

- void KeyDown(KeyEventArgs e)
  Upravuje text *Label*, pokud se jedná o modifikační klávesu.

- void KeyUp(KeyEventArgs e, Shortcuts shorcuts)
  Pokud aplikace čeká na nastavení zkratky, pokusí se ji nastavit. Pokud již existuje daná zkratka, oprostí ji od předešlé funkce a nastaví novou. Pokud je zmáčknut *Esc*, *LWin*, nebo *RWin*, akce se stornuje (vše se vrátí do podoby před nastavováním dané zkratky).

5.10 Pluginy

Pluginy rozeznáváme dvojího typu – vzhledové a efektové. Podle toho jsou definovány jejich rozhraní. Ve svých funkcích mohou obsahovat cokoliv. Pokud jsou v příslušném tvaru, tak je dokáže aplikace spustit, ale neručí za to, co plugin dělá. Pluginy musí být umístěny ve složce „.\Plugins“ a mít koncovku „.dll“.

Rozhraní Plugins.Interfaces.IDesignInterface

Rozhraní pro komunikaci se vzhledovými pluginy.
Struktura:

- string Name
  Vrací nebo nastavuje jméno pluginu.

- Design CreateDesignObject()
  Vrací vzhledový objekt.

Rozhraní Plugins.Interfaces.IEffectInterface

Rozhraní pro komunikaci s efektovými pluginy.
Struktura:

- string Name
  Vrací nebo nastavuje jméno pluginu.

- Effect CreateEffect(AudioStream stream, double from_, double to_)
  Vrací efektový objekt.

5.11 Knihovny

Aplikace používá knihovnu „OpenAL 1.1“, která musí být ve stejné složce jako soubor aplikace a která se musí jmenovat „OpenAL32.dll“.

Pokud se bude používat knihovna verze „1.0“, nepůjde nahrávat (aplikace spadne, protože nenajde danou funkci). Jinak však bude aplikace normálně fungovat. Nezabývám se ošetřením pádu aplikace z důvodu špatné knihovny, jelikož knihovnu požadované verze přikládám.

Jelikož je knihovna psaná v „C++“, používám wrapper. Existují dvě hlavní třídy pro komunikaci s knihovnou, a to:

- static class al
- static class alc

Funkce těchto tříd odpovídají fukcím OpenAL. Ještě existuje třída *ContextAL*, která spravuje globální kontext OpenAL. Obsahuje proměnné (*IntPtr FDevice* a *IntPtr FContext*) a funkce (konstruktor, který inicializuje globální kontext a destruktor, jež ho řádně zruší). Proměnné této řídy jsou potřebné do nějakých funkcí OpenAL. *ContextAL* se vytváří na začátku aplikace a ruší na konci aplikace.

6. Existující implementace a srovnání s nimi

6.1 Existující implementace

Tématem zvukového editoru se zabývá mnoho lidí – ať už komerčně, nebo nekomerčně zaměřených skupin, nebo jednotlivců. Mezi nejoblíbenější rozhodně patří „Audacity“ [17], což je aplikace, která je oblíbená zejména proto, že je open source, multiplatformní (Windows, Linux/Unix, Mac) a existuje pro ní mnoho pluginů. Můj nejoblíbenější editor je „Cool Edit Pro“, který je dnes známý jako „Adobe Audition“ [16] a je zaměřený na platformu Windows. Vyjmenováváním dalších bychom mohli strávit nemalý časový interval, proto jich uvedu jen pár, a to „Wavosaur“ pro Windows, „WavePad“ pro Windows a Mac, „BIAS Peak“ pro Mac, „Jokosher“ pro Linux/Unix atd.

6.2 Srovnání

Profesionální programy na editaci zvuku často mívají složité ovládání nebo jsou velmi nepřehledné. To je většinou dáno velikým počtem funkcí, které ani nejsou potřebné, a jejich nevhodným umístěním. Můj program je oproti těmto programům přehledný. V téměř žádném programu si uživatel nemůže nastavovat klávesové zkratky a jen o málo víc programů si může nastavovat vlastní design. V mém programu je obojí. Velká výhoda jiných implementací je ta, že mají mnohem více efektů a možnost pracovat s jinými formáty zvukových stop. Tato výhoda ovšem způsobuje, že dané aplikace zabírají opravdu hodně místa. Má aplikace nemá s objemem problém, jelikož je malá.

7. Závěr

Vytvořil jsem program, který slouží jako prostředí pro práci se zvukem. Program je přehledný a pohodlný pro používání. Velkou část pohodlnosti ovládání tvoří možnost nastavení vlastních klávesových zkratek a designu. Aplikace je logicky rozdělena na dvě části, a to část, ve které je možné upravovat zvukovou stopu, a část, která umožňuje multitracking. Toto rozdělení výrazně usnadňuje přehlednost celé aplikace.

Mimo různé efekty a editační funkce je zajímavou částí programu spektrální analýza, která rozkládá zpracovávaný signál do frekvenčního spektra.

Komerční zvukové editory často bývají velmi nepřehledné, čímž trpí jednoduchost, můj editor je přehledný.

V programu není zabudováno mnoho efektů, což poněkud omezuje možnosti editace. To je hlavní důvod případného rozšíření, přestože aplikace umožňuje import efektů přes pluginy. Tato omezenost je však i výhodou, protože program netrpí přílišnou velikostí.

Další podněty k možnému rozšíření práce jsou, aby aplikace pracovala i s jinými zvukovými formáty než jen WAVE PCM a aby obsahovala více funkcí v multitrackingové části.

Seznam použité literatury

[1] SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997, ISBN 0-9660176-3-3.
http://www.dspguide.com

[2] VIERS, Ric. *The Sound Effects Bible: How to Create and Record Hollywood Style Sound Effects*. Michael Wiese Productions, 2008, ISBN 9781932907483.
http://books.google.com/books?id=8mocQmOfPz4C

[3] MAREŠ, M. – JAKUBEC, K. – POLÁK, M. – et al. *Fourierova transformace*, 2008.
http://mj.ucw.cz/vyuka/0910/ads2/9-fft.pdf

[4] Převaděč analogového signálu na digitální.
http://en.wikipedia.org/wiki/Analog-to-digital_converter

[5] Interpolace.
http://en.wikipedia.org/wiki/Interpolation

[6] Nejméně významný bit (LSB).
http://en.wikipedia.org/wiki/Least_significant_bit

[7] Struktura souboru WAVE.
http://www.sonicspot.com/guide/wavefiles.html

[8] Shannon-Nyquist vzorkovací teorém.
http://en.wikipedia.org/wiki/Shannon-Nyquist_sampling_theorem

[9] Nyquist frekvence.
http://en.wikipedia.org/wiki/Nyquist_frequency

[10] Leakage.
http://en.wikipedia.org/wiki/Spectral_leakage

[11] Decibel.
http://en.wikipedia.org/wiki/Decibel

[12] Window funkce.
http://en.wikipedia.org/wiki/Window_function

[13] Diskrétní Fourierova transformace.
http://en.wikipedia.org/wiki/Discrete_Fourier_transform

[14] Fourierova transformace a spektrální analýza.
http://www.astro-med.com/knowledge/fourier.html

[15] OpenAL.
http://connect.creativelabs.com/openal

[16] Zvukový editor Adobe Audition.
http://www.adobe.com/products/audition.html

[17] Zvukový editor Audacity.
http://audacity.sourceforge.net/

[18] Mixování zvukových stop.
http://www.vttoth.com/digimix.htm

[19] Vyjádření projektu Mono o WPF.
http://www.mono-project.com/WPF

[20] Overlap-save metoda.
http://en.wikipedia.org/wiki/Overlap-save_method

[21] Archiv zdrojových kódů týkající se zvukových efektů, filtrů atd.
http://www.musicdsp.org/archive.php

Seznam použitých zkratek

FFT = Fast Fourier transform. Rychlá Fourierova transformace.
DFT = Discrete Fourier transform. Diskrétní Fourierova transformace.
LSB = least significant bit. Nejméně významný bit, jednotka používající se pro měření chyby v přesnosti mezi analogovým a digitálním signálem.
WPF = Windows Presentation Foundation. Grafický subsystém pro vykreslování uživatelského rozhraní.
FIR = Finite impulse response. Druh filtru signálu.
GUI = Graphical user interface. Grafické rozhraní aplikace, které slouží ke komunikaci s uživatelem.
MTW = Multitrackingové okno / objekt v aplikaci Wave Editor.
ML = Prvky v multitrackingové části aplikace Wave Editor ovládající vykreslovací kontroly.
RA = Vykreslovací kontroly vykreslující zvukovou stopu v multitrackingové části aplikace Wave Editor.

Příloha A: Uživatelská dokumentace

Wave editor

Wave editor je aplikace, která umožňuje uživateli upravovat zvuk. Dokáže zvuk přehrávat, nahrávat, editovat a mixovat. Umožňuje práci s více stopami. To vše v přizpůsobivém prostředí.

Požadavky

Aplikace funguje na operačních systémech *Windows Vista* a novějších. Též funguje na *Windows XP SP2/SP3* a *Windows Server 2003* – na těchto systémech musí být ovšem instalovány knihovny *.NET Framework 3.0*.

Instalace

Stačí přesunout nebo zkopírovat složku s programem včetně jeho podadresářů na cílové místo.

Spuštění

Aplikace se spouští souborem „WE.exe“.

Popis prostředí

Aplikace se skládá ze dvou oken – editačního a multitrackingového. Editační okno je určeno pro editaci zvukové stopy, zatímco multitrackingové okno je určeno pro práci s více stopami.

Editační okno

Toto okno slouží k editování zvukové stopy. Rozložení prvků v okně je znázorněno na obrázku 7.1. Popis hlavních prvků je uveden níže:

- Menu
  Menu, pomocí kterého uživatel komunikuje s aplikací.
- Zobrazovací část
  Do této části je vykreslována zvuková stopa.
- Info o zobrazení
  Obsahuje informace o zobrazené části stopy a jejím označení.
- Panel přehrávání
  Panel sloužící k ovládání přehrávání a nahrávání.

Obrázek 7.1: Editační okno

- Info o stopě
  Bližší informace o zvukové stopě. Např. rozlišení, délka, počet kanálů.

Multitrackingové okno

Okno je určeno pro práci s více stopami. Stopy v tomto okně však nelze editovat. Lze je vkládat, mazat, posouvat, přehrávat atd. Rozložení prvků v okně je znázorněno na obrázku 7.2. Popis hlavních prvků je uveden níže:

- Menu
  Menu, pomocí kterého uživatel komunikuje s aplikací.
- Zobrazovací část
  Do této části jsou vykreslovány zvukové stopy.
- Panel přehrávání
  Panel sloužící k ovládání přehrávání.

Ovládání

Přepínání oken

Uživatel přepne okna stisknutím tlačítka „SWITCH“.

Obrázek 7.2: Multitrackingové okno

Editační okno

Načtení a ukládání zvukové stopy

- Načtení zvukové stopy
  Uživatel může načíst zvukovou stopu z disku přes ovládací menu (File –> Load) nebo přes základní zkratku. Otevře se rozhraní pro otevření souboru. Základní zkratka je nastavena na „Ctrl + O“. *Pozn. Soubor musí být ve formátu WAVE PCM(.wav). Rozlišení stopy musí být 8-bit, nebo 16-bit, počet kanálů mono, nebo stereo.*

- Uložení zvukové stopy
  Uživatel může uložit zvukovou stopu na disk přes ovládací menu (File –> Save) nebo přes základní zkratku. Otevře se rozhraní pro uložení souboru. Základní zkratka je nastavena na „Ctrl + S“.

- Vyčištění okna
  Uživatel může oprostit editační okno od zobrazované stopy přes ovládací menu (File –> Clear) nebo přes základní zkratku. Základní zkratka je nastavena na „Ctrl + Del“.

Zobrazování

Aktuální stopa se vykresluje v zobrazovací části. Aplikace nabízí uživateli řadu zobrazovacích funkcí, které jsou umístěny v Menu –> View. Samozřejmě je možné tyto funkce pouštět i přes klávesové zkratky.

- Přiblížení stopy
  V menu lze přiblížit stopu přes položku „Zoom in“. Základní zkratka je „Ctrl + +“. Přibližovat lze též kolečkem myši – stopa se přibližuje do místa, kde je kurzor myši.

- Oddálení stopy
  V menu lze oddálit stopu přes položku „Zoom out“. Základní zkratka je „Ctrl + -“. Oddalovat lze též kolečkem myši – stopa se oddaluje od místa, kde je kurzor myši.

- Přiblížení výběru
  Pokud je označena část stopy, lze se přiblížit do tohoto označení přes položku „Zoom selection“

- Zobrazení celé stopy
  V menu reprezentováno položkou „Zoom all“.

- Vertikální roztáhnutí resp. stáhnutí zobrazování
  Roztáhne resp. stáhne amplitudu zvukové stopy. Položka „Zoom in (vertically)“ resp. „Zoom out (vertically)“.

- Posouvání stopy
  Stopu lze posouvat scrollbarem nebo přes položky „Move left“ a „Move right“

Označování

Pokud je zobrazována nějaká zvuková stopa, může ji uživatel označovat stisknutím a tahem levého tlačítka myši nebo případně označit vše klávesovou zkratkou „Ctrl + A“.

Přehrávání a nahrávání

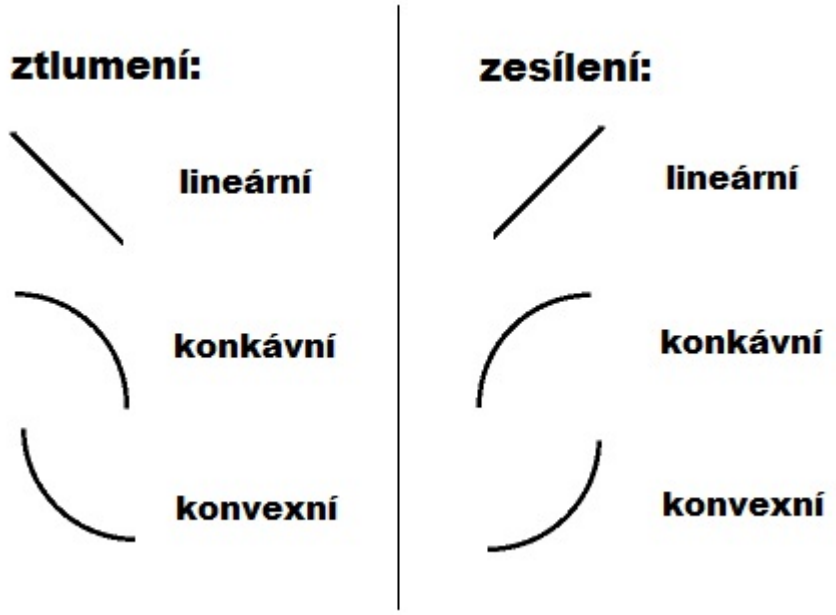
Tlačítka pro ovládání přehrávání a nahrávání jsou umístěna v panelu přehrávání. Aplikace reaguje i na klávesové zkratky. Uživateli aplikace nabízí následující akce:

- Spuštění přehrávání
  Začne přehrávat zvukovou stopu v závislosti na označení. Reprezentováno tlačítkem „Play“. Původní zkratka je „Q“.

- Pauza resp. odpauzování přehrávání
  Pozastaví přehrávání i nahrávání, nebo v nich pokračuje. Reprezentováno tlačítkem „Pause“ resp. „Unpause“. Původní zkratka je „W“.

- Zastavení přehrávání
  Zastaví přehrávání, nebo nahrávání. Reprezentováno tlačítkem „Stop“. Původní zkratka je „E“.

- Nahrávání
  Spustí nahrávání stopy. Před tím ještě zobrazí dialog, kde si uživatel zvolí parametry záznamu. Reprezentováno tlačítkem „Record“. Původní zkratka je „R“. Po nahrávání zobrazí do editačního okna právě nahranou stopu.



Obrázek 7.3: Postupné zesílení a ztlumení

Editace

Všechny editace může uživatel vzít zpět akcí „Undo“(Menu –> Edit –> Undo) resp. „Redo“(Menu –> Edit –> Redo).

- Stříhání
  Aplikace nabízí akce kopírování, vkládání zkopírované části, mazání, vyjmutí a oříznutí. Tyto akce závisí na označení stopy. Spuštění těchto funkcí uživatel nalezne v Menu –> Edit. Akce mají také klávesové zkratky – Ctrl+C (kopírování), Ctrl+V (vložení), Del (mazání), Ctrl+X (vyjmutí) a Ctrl+T (oříznutí).

- Úpravy hlasitosti
  Aplikace nabízí uživateli akce, které hlasitostně upraví označený úsek.

  – Zesílení
    Zesílí označenou část. Uživatel může spustit akci přes Menu –> Volume –> Up. Lze nastavit klávesovou zkratku.
  – Ztlumení
    Ztlumí označenou část. Uživatel může spustit akci přes Menu –> Volume –> Down. Lze nastavit klávesovou zkratku.
  – Postupné zesílení
    Postupně zesílí označenou část. Na výběr má uživatel 3 možnosti (viz 7.3), ke kterým se dostane přes Menu –> Volume –> Fade up.
  – Postupné ztlumení
    Postupně ztlumí označenou část. Na výběr má uživatel 3 možnosti (viz 7.3), ke kterým se dostane přes Menu –> Volume –> Fade down.

- Efekty a filtry
  Efekty i filtry transformují zvukovou stopu v závislosti na označení. Uživatel je může aplikovat na stopu přes panel efektů nebo filtrů. Viz níže.

Konkrétní zvukové efekty a filtry

- Průměrující efekt
  Tento filtr vyhlazuje náhlé výkyvy ve zvukové stopě. Spolu s tím ovšem vyhlazuje i vyšší frekvence. Kolik vzorků vedle sebe se bude průměrovat, udává parametr „Samples“. Filtr uživatel najde v panelu filtrů pod názvem „Average filter“.

- Frekvenční filtr
  Frekvenční filtr zesiluje a zeslabuje určité frekvence v závislosti na parametrech. Filtr uživatel najde v panelu filtrů pod názvem „Frequency filter“.

- Ozvěna
  Efekt ozvěny způsobí ozvěnu závislou na parametrech. Parametr „Delay“ udává zpoždění zvuku a parametr „Fade down“ udává míru utlumení zvuku. Efekt uživatel najde v panelu efektů pod názvem „Delay“.

- Překlopení zvuku
  Tento efekt daný zvuk překlopí tak, že je slyšet pozpátku. Efekt uživatel najde v panelu efektů pod názvem „Reverse effect“.

Spektrální analýza

Po spuštění akce v Menu –> Analyze –> Spectral analysis se zobrazí uživateli okno, ve kterém je vykreslen spektrogram dané stopy. Pro větší přesnosti může uživatel měnit velikost bufferu FFT (čím větší, tím přesnější spektrogram) a zobrazovací funkce. Uživatel si též může zvolit styl měřítka spektrogramu. Výsledný obrázek si může uživatel uložit na disk.

Multitrackingové okno

Načítání a mazání zvukových stop

- Vložení zvukové stopy
  Uživatel může vložit zvukovou stopu z disku nebo již nahranou stopu, pokud klikne pravým tlačítkem myši na příslušné místo v zobrazovací části a zvolí „Insert“. Pokud dále zvolí „Wave from file...“, otevře se rozhraní pro načtení stopy z disku. Daná stopa se vloží na příslušné místo. *Pozn. Soubor musí být ve formátu WAVE PCM(.wav). Rozlišení stopy musí být 8-bit, nebo 16-bit, počet kanálů mono, nebo stereo.*

- Mazání zvukové stopy
  Uživatel může vymazat zvukovou stopu ze zobrazovací části tím, že na ni klikne pravým tlačítkem myši a zvolí „Remove“.

- Vyčištění okna
  Uživatel může oprostit multitrackingové okno od zobrazovaných stop přes

ovládací menu (File –> Clear) nebo přes základní zkratku. Základní zkratka je nastavena na „Ctrl + Del“.

Zobrazování

Aktuální stopy se zobrazují v zobrazovací části. Aplikace nabízí uživateli zobrazovací akce, které jsou umístěny v Menu –> View. Tyto funkce jsou „přiblížení“ a „oddálení“. Je možné tyto funkce pouštět i přes klávesové zkratky. Pohled na zvukové stopy lze posouvat scrollbarem.

Označování

Uživatel může označovat úseky v zobrazovací části stisknutím a tahem levého tlačítka myši.

Posouvání

Uživatel může posouvat stopy v zobrazovací části stisknutím a tahem pravého tlačítka myši na danou stopu.

Přehrávání

Tlačítka pro ovládání přehrávání jsou umístěna v panelu přehrávání. Aplikace reaguje i na klávesové zkratky. Aplikace nabízí uživateli následující akce:

- Spuštění přehrávání
  Začne přehrávat zvukové stopy v závislosti na označení. Reprezentováno tlačítkem „Play“. Původní zkratka je „Q“.
- Pozastavení resp. odpauzování přehrávání
  Pozastaví přehrávání, nebo v něm pokračuje. Reprezentováno tlačítkem „Pause“ rep. „Unpause“. Původní zkratka je „W“.
- Zastavení přehrávání
  Zastaví přehrávání. Reprezentováno tlačítkem „Stop“. Původní zkratka je „E“.

Nastavení

Uživatel může nastavit klávesové zkratky a design daného okna. K nastavení se dostane přes Menu –> Option –> Settings. Tato akce spustí dialog pro nastavení klávesových zkratek a designu. Ovládání je intuitivní. Při nastavování zkratek se nedoporučuje používat Space a Enter. Neměla by se používat též klávesa Alt. Při nastavování vzhledu může uživatel zaškrtnout, jestli chce, aby byl design jednotný pro celou aplikaci.

V editačním okně si může uživatel nastavit viditelnost panelů přes Menu –> Option –> Toolbars.

Pluginy

Uživatel může používat akce pluginů (Menu–>Plugins). Tyto akce buď změní vzhled, nebo vykonají efekt. Co pluginy ve skutečnosti dělají nemůže aplikace zaručit, proto je doporučeno používat pluginy pouze z věrohodných zdrojů. Efektové pluginy existují pouze v editační části a editují stopu v závislosti na označení (můžou si definovat vlastní prostředí pro komunikaci s uživatelem).

Příloha B: CD

- Text bakalářské práce v digitální podobě je v „\WaveEditor.pdf“.
- Zdrojové kódy programu jsou ve složce „\Kody“.
- Aplikace je umístěna ve složce „\WaveEditor“ a spouští se souborem „WE.exe“.